AD_______________

Award Number: DAMD17-98-1-8044

TITLE: A Computer-Based Decision Support System for Breast Cancer Diagnosis

PRINCIPAL INVESTIGATOR: Zuyi Wang, Ph.D.

CONTRACTING ORGANIZATION: The Catholic University of America
Washington, DC 20064

REPORT DATE: September 2000

TYPE OF REPORT: Annual Summary

PREPARED FOR: U.S. Army Medical Research and Materiel Command
Fort Detrick, Maryland 21702-5012

20010511 148

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 074-0188*

1. AGENCY USE ONLY (Leave blank)	**2. REPORT DATE** September 2000	**3. REPORT TYPE AND DATES COVERED** Annual Summary (1 Sep 99 - 31 Aug 00)

4. TITLE AND SUBTITLE A Computer-Based Decision Support System for Breast Cancer Diagnosis	**5. FUNDING NUMBERS** DAMD17-98-1-8044
6. AUTHOR(S) Zuyi Wang, Ph.D.	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) The Catholic University of America Washington, DC 20064 E-MAIL: zwang@pluto.ee.cua.edu	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) U.S. Army Medical Research and Materiel Command Fort Detrick, Maryland 21702-5012	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution unlimited	**12b. DISTRIBUTION CODE**

13. ABSTRACT ***(Maximum 200 Words)***

In the second year of the project, we devoted efforts in developing effective feature extraction methods, constructing feature database, developing visual explanation tool for data mining and knowledge discovery, which is both statistically principled and visually effective. This method, as illustrated by the well-planned simulations and pilot applications in computer-aided diagnosis, can be very capable of revealing hidden structure within data. It is important to emphasize that the present algorithm is that the models are determined by the information theoretic criteria, and this criterion can not only select the most appropriate model structure but also allow a user-driven portfolio as a double check. In addition, since we perform model selection and parameter initialization firstly over the projection space, the computational complexity is greatly reduced in compared to the maximum likelihood estimation in full dimension. Other possible advantages include the determination of data projection by maximum the separation of clusters which in turn optimizes the other crucial operations such as model selection and parameter initialization, and data rendering algorithms which permit user or hypothesis driven nature of the data projection. Using the visual explanation tool, we are trying to discover the feature database structure for case, feature selection and classifier design.

14. SUBJECT TERMS Breast Cancer Diagnosis, Breast Cancer Patient Database, Decision Support System, Computer-Aided Diagnosis, visual explnantion, Artificial Intelligence	**15. NUMBER OF PAGES** 58
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	Unlimited

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

Table of Contents

A Computer-Based Decision Support System for Breast Cancer Diagnosis

Introduction

The goal of this project is to develop decision support system for breast cancer diagnosis, treatment option, prognosis, and risk prediction. This system is desired to function as a consultation system for both doctors and patients. This pre-doctoral research project is focusing on the development of advanced image pattern analysis in diagnostic imaging and information integration methodology. The specific aims of this research project are: (1) image pattern analysis of breast tissue in mammography using both computational features and BI-RADS features provided by radiologist for the prediction of malignancy associated with masses; (2) development of visual presentation methods for radiologists' use in the consultation system; (3) performing a pre-clinical test through an ROC analysis. The clinical goal of this consultation system is to provide scientific tools for doctors to have electronic magnification views, to perform feature analysis of suspected mammographic patterns, to access a large database and investigate clinically similar cases, and to visually inspect the features of a case in various statistical distribution using graphic displays. The primary objectives of the research in the second year are feature extraction, feature database construction, visual explanation tools development, and feature database structure exploration using visual explanation tool.

Overview of Training and Research Accomplishment of Phase II

1. Research Skill Training and Literature Background Preparation

In the past year, trainings on research skill gained through working with the mentors of this project are very important to the whole research process, which made it possible for me to continue my research work. From the first program for reading and processing digital mammogram, to the selection of cases, and then to the understanding of fundamental engineering components that are essential the project development, my academic advisor Dr. Yue Wang at The Catholic University of America, my mentors Dr. Shih-Chung Lo and Dr. Matthew Freedmen at Georgetown University provided so much tremendous help as they can. After one year of research work, my insight on research approach and ability of problem solving has been gradually established and improved. We often discussed and reviewed the primary goal of this project in the research process in order to keep my work in the right direction and give a global view of the all components of CAD. They helped me write more practical programs for image processing, and discuss the intermediate results of calculation with me for further research planning.

Under the guidance of Dr. Wang and Dr. Lo, literature and book searching and reading gave me better and broader view of breast cancer and computer-assisted diagnosis (CAD) system research. Through reading engineering textbooks, the fundamental knowledge that is critical to the project is greatly enhanced. The major books I have been reading and using as all-time references are *Neural Network - A Comprehensive foundation* by Simon Haykin, *An Introduction to Signal Detection and Estimation* by H. Vincent Poor, and *Elements of Information Theory* by Thomas M. Cover, etc.. After searching and technical papers in several major engineering journals, such as IEEE Transactions on Medical Imaging, IEEE Transactions on Neural Network, IEEE Transactions on Pattern Recognition and Machine Intelligence, and Medical Physics, etc., I have collected almost one hundred of relevant papers for myself to know works done by other researchers and also set a start point and direction for my own research. The more I read, the better my ability of understanding and judging others' work.

2. Research Accomplishments

2.1 Clinical Case and Feature Database Development

2.1.1 Clinical Case Selection

As the first step for establishing a feature database, case searching and selection are fundamental and crucial for the further research work. In order to detect suspicious mass regions from a mammogram, we have to be able to find out the major differences between mass and non-mass regions so that both real mass and non-mass case groups are needed for comparison purpose. Two major mammogram sample sources are found proper for the use in this project, one is ISIS at Georgetown University Medical Center, the other is Mammographic Image Analysis Society (MIAS) web site that is open for public research use. The ISIS database is constructed by extracting suspicious mass regions from mammograms by licensed radiologists and finally proven by biopsy procedure, from where we obtained 125 cases, among these 75 are mass cases and 50 are non-mass cases. Non-mass cases were selected from normal breast tissue regions with similarity of mass. In the web site of MIAS, we found the 36 mass cases and 40 non-mass cases. The total number of cases that are currently used in the project is 201, and more cases might be added if needed. All cases are extracted from the whole mammograms as image blocks for further feature calculation use.

2.1.2 Image Feature Extraction

After the preparation of mammogram cases, the next important consideration is to choose features that can be used to distinguish mass and non-mass cases effectively and with high detective rate. The image block was first processed by enhanced segmentation procedure to extract the exact position where a mass may present. The position of the segmented area was then a very useful reference for feature calculation. Many features have been tested by other researchers for effectiveness, and the results have been presented in their most recent papers. Based on literature and medical book searching and reading, primarily we decided on ten features, among them are eight texture features, shape feature and margin feature. Eight texture features were calculated based spatial gray level dependence matrix, they are energy, correlation, inertia, entropy, Inverse difference moment, sum average, sum entropy, and difference entropy. Texture feature, in some scale, may be fairly good for reveal fine texture differences in images, which human eyes have difficulty to see. They were examined by several research groups for effectiveness in terms of improvement of CAD performance.

Shape feature, such as compactness has been used to exclude non-mass cases from the whole case population in previous study. By observation of hundreds mammograms, shape feature is found to be essential for detecting a mass merging in many mass-like normal breast tissues. Most of masses have relatively well-defined round object in the center regardless to the speculated margin, however, the overall shape of dense normal breast tissues, such as glandular elements and blood vessels embedded, are often slender with no well-defined object in the center. The simplest way of compactness calculation is to divide the area of the segmented area by the perimeter of the contour. The method was found not very effective in those cases with rippled contours since two segmented areas with same shape could have different compactness if one of them has more rippled contour. The shape calculation method that can eliminate the effect of rippled contour has been improved to replace the simple compactness feature. The new method can automatically search the long axis of the segmented area for initialization, do nonlinear fitting on the contour based on Levenberg-Marquardt method, and eventually separate shapes in more accurate scale from very round to a line. Levenberg-Marquardt method works well in practical has become the standard of nonlinear least-square routines for varying smoothly between the extremes of the inverse-Hessian method and the steepest descent method.

Margin feature has been recognized as the only reliable specific feature used by radiologists to decide whether a mass is malignant or benign, and also it is one of the most important features that can help mass recognition from normal breast tissue. The difficulty for effectively and precisely extracting margin feature that truly belongs to the object becomes greater due to the overlapping of normal breast tissue on the object region makes a smooth margin speculated, thus recognizing true spicules is a very challenging task we are attempting.

3. Feature Database Explanation Using Visual Data Explanation and Mining Tool

3.1.1 Visual Data Explanation and Mining Tool Development

Although among many approaches of CAD research, some CAD systems are sophisticated and claimed to have impressive performance, several fundamental issues remain unsolved. For example, Receiver Operating Characteristics (ROC) can provide an overall performance evaluation, but it may not help improve each individual component in CAD system. Furthermore, since machine observer and human observer may not detect the same set of masses, the black box nature of most CAD systems may prevent a natural on-line integration of human and machine intelligence and further upgrade of a CAD system. As a strategic move toward improving CAD design and utility, we developed a visual data explanation and mining tool. Our effort is to (1) provide a visual map of feature database prior to knowledge encoding component so as to evaluate and improve the pre-processing and signature extraction; (2) based on the resulting map to design an optimal classifier best fitted to the particular database structure for knowledge encoding; and (3) combine the map, the classifier output, raw image, and user interface to explore and explain the whole decision making process by both radiologist and CAD system.

The revelation of growing volume of high dimensional and multi-modal data set demands a data mining tool differing from conventional data visualization method, which is capable of dealing with high dimensional data set. This motivates our consideration of a hierarchical visualization paradigm involving hierarchical statistical models and visualization space. Comprehensive studies on this issue brought us the possibility of using several complementary visualization subspaces to accomplish this complicated task. In this algorithm, dimensionality reduction and cluster decomposition are two major components. The cluster decomposition permits the use of relatively simple models for each local structure, offering great ease of interpretation as well as many benefits of analytical and computational simplification. On the other hand, dimensionality reduction allows visual explanation of high dimensional data set and less computational demand. We proposed using standard finite normal mixtures (SFNM) and hierarchical visualization spaces for as effective data modeling and visualization. The strategy is that top level model and projection should explain the whole structure of the data set, while lower level models explain the local and internal structure between individual cluster, which may not be obvious in the high level models. With many complementary mixture models and visualization projections, each level will be relatively simple while the complete hierarchy maintains overall flexibility yet still conveys considerable cluster information. Based on the concept of combining finite mixture modeling and principal component projection to guide cluster decomposition and dimensionality reduction, the particular advantages of our method are: (1) at each level, a probabilistic principle component analysis is performed to project the softly partitioned data space down to a desired two-dimensional visualization space, leading to an optimal dimensionality reduction that allows best separation and visualization of local clusters; (2) learning from the data directly, minimax entropy principle is used to select model structures and estimate its parameter values, where the soft partitioning of the data set results in a standard finite normal mixture model with minimum conditional bias and variance; (3) by performing principal component analysis (PCA) and minimax entropy modeling alternatively, a complete hierarchy of complementary projections and refined models can be generated automatically, corresponding to a statistical description best fit to the data.

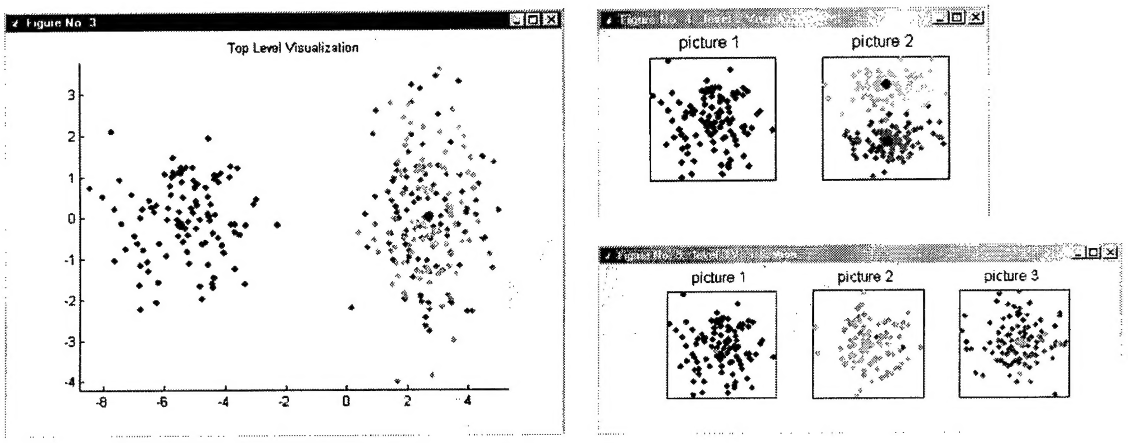


Figure 1. User Interface of Visual Data Explanation and Mining Tool

User interaction with the algorithm is also an important issue. We have developed a user-friendly graphical interface to facilitate the data visualization purpose, as shown in Fig. 1, which allows the user to select initial centers of the data clusters. Our experience has convincingly indicated a great reduction of both computational complexity and local optimum likelihood. It should be pointed out that although the final SFNM model can be estimated, the pathways of achieving cluster decomposition may be multiple. For example, in this case the user has the flexibility to select only two clusters in the second level and to further split the ``right" cluster, thus to adopt a three-level hierarchy. We believe that this user-driven nature of the current algorithm is also highly appropriate for the visualization context.

3.1.2 Feature Database Explanation for Case and Feature Selection, and Classifier Design Purpose

As the primary goal of the visual explanation and mining tool development, we use it to reveal and explain feature database structure for CAD design purpose. We try to make both hidden data patterns and neural network "black box" to be as transparent as possible to users, such as radiologists and patients through interactive visual explanation. The clinical purpose is to eliminate the false positive sites that correspond to normal dense tissue with mass-like appearances through feature discrimination. As described in Image Feature Extraction section, we adopted mathematical feature extraction procedures to construct our feature database based on all the suspicious mass sites localized by our enhanced segmentation method. We then put efforts in exploring feature database structure using the visual data explanation and mining tool in order to make the design processes of some components in CAD more reasonable and efficient so as to improve CAD performance. Cluster modeling gives case selection a reliable base that may help select cases truly representing mass and non-mass categories. We believe that using the cases selected carefully and reasonably in such way for classifier training may further make classifier training more effective so as to improve classifier performance.

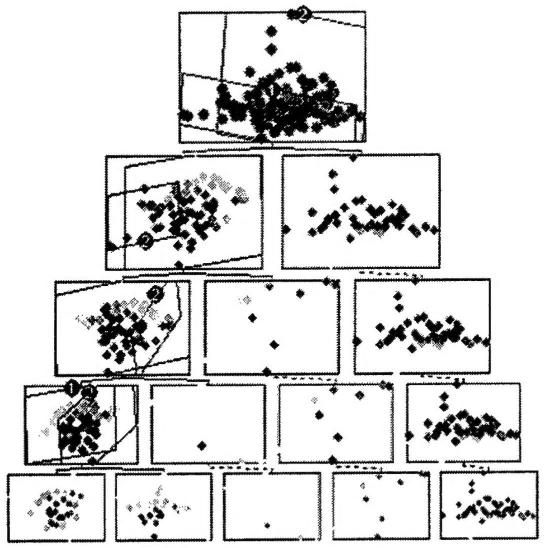


Figure 2. Hierarchical view of nine features for mass and non-mass cases

Data exploration using our visual use not only help case selection, also is beneficial to the evaluation and selection of feature. By putting efforts in observing tens of combinations of feature, interim results show that shape and some texture features, such as energy, correlation and difference entropy, are better distinguishable between mass and non-mass categories in term of farther separation of data points and better defined data structure. Margin feature will be tested when soon being ready. In classifier selection and design, feature database structure is the major guidance we can depend on, and this work has been started. A hierarchical view of nine-feature database is show in Fig. 2 as an example of revealing data structure. All these approaches have the only important goal that is to improve CAD performance in a rational way so that we can explain how we design each component of the CAD system, why such a integrated system works or does not work, and further explain to radiologists to get feedback on the development, the process is fairly transparent to users.

Key Research Accomplishments

- Improving research skill and enhancing fundamental engineering knowledge through book and literature searching and reading under guidance of advisor and mentors.
- Collecting image cases, processing images by computing image features and constructing high dimensional image feature database.
- Developing visual data explanation and mining tool and exploring feature database structure for case feature selection and classifier design to make the CAD design processing effective and reasonable.

Reportable Outcomes

- Y. Wang, Z, Wang, L. Luo, S-H. B. Lo and M. T. Freedman, "Computer-Based Decision Support System: Visual Mapping of Featured Database in Computer-Aided Diagnosis", *Proc. Of SPIE, Image Processing,* Vol. 1, No. 24, pp. 136-147, February 2000.
- Feature extraction programs.
- Visual data explanation and mining tool software.

Conclusions

In the second year of the project, we devoted efforts in developing effective feature extraction methods, constructing feature database, developing visual explanation tool for data mining and knowledge discovery, which is both statistically principled and visually effective. This method, as illustrated by the well-planned simulations and pilot applications in computer-aided diagnosis, can be very capable of revealing hidden structure within data. It is important to emphasize that in relation to previous work, one interesting consideration with the present algorithm is that the models are determined by the information theoretic criteria, and this criterion can not only select the most appropriate model structure but also allow a user-driven portfolio as a double check. This approach promotes a self-consistent fitting of the whole tree, so that an automated procedure for generating the hierarchy becomes reality. In addition, since we perform model selection and parameter initialization firstly over the projection space, the computational complexity is greatly reduced in compared to the maximum likelihood estimation in full dimension. Other possible advantages include the determination of data projection by maximum the separation of clusters, which in turn optimizes the other crucial operations such as model selection and parameter initialization, and data rendering algorithms which permit user or hypothesis driven nature of the data projection. Using the visual explanation tool, we are trying to discover the feature database structure for case and feature selection and also classifier design.

References

1. Y. Wang, Z, Wang, L. Luo, S-H. B. Lo and M. T. Freedman, "Computer-based decision support system: visual mapping of featured database in computer-aided diagnosis", *Proc. Of SPIE, Image Processing,* Vol. 1, No. 24, pp. 136-147, February 2000.
2. Y. Wang, L. Luo, M. T. Freedman and S-Y. Kung, "Probabilistic principal component subspaces: a hierarchical finite mixture model for data visualization", *IEEE Trans. on Neural Networks,* Vol. 11, No. 3, pp. 625-636, May 2000.
3. C. M. Bishop and M. E. Tipping, ``A hierarchical latent variable model for data visualization," *IEEE Trans. Pattern Anal. Machine Intell.*, Vol. 20, No. 3, pp. 282-293, March 1998.
4. L. Luo, Y. Wang, and S. Y. Kung, ``Hierarchy of probabilistic principal component subspaces for data mining," *Proc. IEEE Workshop on Neural Nets for Signal Processing*, Wisconsin, August 1999.
5. T. M. Cover and J. A. Thomas, *Elements of Information Theory*, New York: Wiley, 1991.

```
/*

    This program is designated for calculating texture feature of
    ROIs in mammograms.
    - Inside and outside of segmented areas are both calculated.
    - Run all images at once.

    Zuyi Wang

*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <memory.h>
#include <io.h>

#define  GRAY_LEVEL 256


void SGLD_theda_d(int d, float *SGLD_ptr, float *SGLD_out_ptr);
void print_SGLD_matrix(int d, float *SGLD_ptr);
void write_output_file(double energy_tf, double correlation_tf, double inertia_tf, double e
ntropy_tf,
    double inv_diff_moment_tf, double sum_average_tf, double sum_entropy_tf,double diff_ent
ropy_tf, double even_tf);
void verify_SGLD_matrix(float *SGLD_ptr);
double energy(int d, float *SGLD_ptr);
double correlation(int d, float *SGLD_ptr);
double inertia(int d, float *SGLD_ptr);
double entropy(int d, float *SGLD_ptr);
double inv_diff_moment(int d, float *SGLD_ptr);
double sum_average(int d, float *SGLD_ptr);
double sum_entropy(int d, float *SGLD_ptr);
double diff_entropy(int d, float *SGLD_ptr);
double even(int d, float *SGLD_ptr);


char in_filename[100] ;
char mor_filename[100];
char temp_filename[100];
char output_filename[100];

void main(int argc, char* argv[])
{
    if ( argc != 2 )
    {
        printf("No file in this directory!!!\n");
        exit(1);
    }

    strcpy(in_filename, argv[1]);

    FILE *output_fptr;

    int d = 1; //Distance -- number of pixels
    float *SGLD_ptr, *SGLD_out_ptr;
    float SGLD_matrix[GRAY_LEVEL][GRAY_LEVEL] = {0};
    float SGLD_matrix_out[GRAY_LEVEL][GRAY_LEVEL] = {0};
    double energy_tf, correlation_tf, inertia_tf, entropy_tf;
    double inv_diff_moment_tf, sum_average_tf, sum_entropy_tf, diff_entropy_tf/*, even_tf*/
;
    double energy_out_tf, correlation_out_tf, inertia_out_tf, entropy_out_tf;
    double inv_diff_moment_out_tf, sum_average_out_tf, sum_entropy_out_tf, diff_entropy_out
_tf/*, even_out_tf*/;


    SGLD_ptr = &SGLD_matrix[0][0];
    SGLD_out_ptr = &SGLD_matrix_out[0][0];

    SGLD_theda_d(d, SGLD_ptr, SGLD_out_ptr);

    /* Calculate texture for segmented area */
    energy_tf = energy(d, SGLD_ptr);
    correlation_tf = correlation(d, SGLD_ptr);
    inertia_tf = inertia(d, SGLD_ptr);
    entropy_tf = entropy(d, SGLD_ptr);
    inv_diff_moment_tf = inv_diff_moment(d, SGLD_ptr);
    sum_average_tf = sum_average(d, SGLD_ptr);
    sum_entropy_tf = sum_entropy(d, SGLD_ptr);
    diff_entropy_tf = diff_entropy(d, SGLD_ptr);
    //even_tf = even(d, SGLD_ptr);

    /* Calculate texture for outside of segmented area */
    energy_out_tf = energy(d, SGLD_out_ptr);
    correlation_out_tf = correlation(d, SGLD_out_ptr);
    inertia_out_tf = inertia(d, SGLD_out_ptr);
    entropy_out_tf = entropy(d, SGLD_out_ptr);
    inv_diff_moment_out_tf = inv_diff_moment(d, SGLD_out_ptr);
    sum_average_out_tf = sum_average(d, SGLD_out_ptr);
    sum_entropy_out_tf = sum_entropy(d, SGLD_out_ptr);
    diff_entropy_out_tf = diff_entropy(d, SGLD_out_ptr);
    //even_out_tf = even(d, SGLD_out_ptr);

    //print_SGLD_matrix(2, SGLD_ptr);

    printf("%s\n%s %f\n%s %f\n%s %f\n%s %f\n%s %f\n%s %f\n%s %f\n%s %f\n",
            "Texture features computed for segmented area:",
            "Energy = ", energy_tf,
            "Correlation =", correlation_tf,
            "Inertia = ", inertia_tf,
            "Entropy = ", entropy_tf,
            "Inverse difference moment = ", inv_diff_moment_tf,
            "Sum average = ", sum_average_tf,
            "Sum entropy = ", sum_entropy_tf,
            "Difference entropy = ", diff_entropy_tf);

    printf("\n%s\n%s %f\n%s %f\n%s %f\n%s %f\n%s %f\n%s %f\n%s %f\n%s %f\n",
            "Texture features computed for outside of segmented area:",
            "Energy = ", energy_out_tf,
            "Correlation =", correlation_out_tf,
            "Inertia = ", inertia_out_tf,
            "Entropy = ", entropy_out_tf,
            "Inverse difference moment = ", inv_diff_moment_out_tf,
            "Sum average = ", sum_average_out_tf,
            "Sum entropy = ", sum_entropy_out_tf,
            "Difference entropy = ", diff_entropy_out_tf);
    //verify_SGLD_matrix(SGLD_ptr);
    /*write_output_file(energy_tf, correlation_tf, inertia_tf, entropy_tf, inv_diff_moment_
tf,
        sum_average_tf, sum_entropy_tf,diff_entropy_tf, even_tf);*/

    /* Write results into output file out.txt */
    strcpy(output_filename, "out.txt");

    if ((output_fptr = fopen(output_filename, "a")) == NULL) {
        printf("Cannot open output file %s \n",output_filename);
        exit(1);
    }


    fprintf(output_fptr, "\n%s\n", in_filename);
/*  fprintf(output_fptr, "%s\n", mor_filename);
*/
    /*fprintf(output_fptr, "%s\n%s %f\n%s %f\n%s %f\n%s %f\n%s %f\n%s %f\n%s %f\n%s %f\n%s
%f\n",
            "Texture features computed for segmented area:",
            "Energy = ", energy_tf,
            "Correlation =", correlation_tf,
            "Inertia = ", inertia_tf,
            "Entropy = ", entropy_tf,
            "Inverse difference moment = ", inv_diff_moment_tf,
            "Sum average = ", sum_average_tf,
            "Sum entropy = ", sum_entropy_tf,
            "Difference entropy = ", diff_entropy_tf,
            "Even = ", even_tf);

    fprintf(output_fptr, "%s\n%s %f\n%s %f\n%s %f\n%s %f\n%s %f\n%s %f\n%s %f\n%s %f\n%s %f
\n",
            "Texture features computed for outside of segmented area:",
            "Energy = ", energy_out_tf,
            "Correlation =", correlation_out_tf,
            "Inertia = ", inertia_out_tf,
            "Entropy = ", entropy_out_tf,
            "Inverse difference moment = ", inv_diff_moment_out_tf,
            "Sum average = ", sum_average_out_tf,
            "Sum entropy = ", sum_entropy_out_tf,
            "Difference entropy = ", diff_entropy_out_tf,
            "Even = ", even_out_tf);
*/
    /*
    Write result of inside and outside of mass in two separate files --- run program twice
    inside result
    */
    fprintf(output_fptr, "%f\n%f\n%f\n%f\n%f\n%f\n%f\n%f\n",
            energy_tf, correlation_tf, inertia_tf, entropy_tf,
            inv_diff_moment_tf, sum_average_tf, sum_entropy_tf,diff_entropy_tf);

    //outside result
    /*fprintf(output_fptr, "%f\n%f\n%f\n%f\n%f\n%f\n%f\n%f\n",
            energy_out_tf, correlation_out_tf, inertia_out_tf, entropy_out_tf,
            inv_diff_moment_out_tf, sum_average_out_tf, sum_entropy_out_tf,diff_entropy_out
_tf);
    */

    fclose(output_fptr);
```

```
}


void SGLD_theda_d(int d, float *SGLD_ptr, float *SGLD_out_ptr)
{
    int i, j, temp_i, temp_j, gray_a, gray_b, gray_a_mor, gray_b_mor;
    int a, b, theda, total_pix_pair = 0, total_pix_pair_out = 0;
    int fhandle, fhandle_mor;
    int hn, hn_mor, ROI_X, ROI_Y, ROI_X_MOR, ROI_Y_MOR;

    long ROI_size, ROI_size_mor;
    unsigned short header[128], header_mor[128];
    unsigned char *img_buffer, *ROI, *img_buffer_mor, *ROI_MOR;


    FILE *fptr;
    FILE *fptr_mor;

//  Read filename
//  printf("Please enter ROI filename: ");
//  gets(in_filename);

    /*
    Open ROI file
    */
    if ((fptr = fopen(in_filename,"rb")) == NULL) {
        printf("Cannot open %s \n",in_filename);
        exit(0);
    }
    else {
        printf("Original image file is %s\n", in_filename);
    }


    //Check if file extension is .mor (false mass) or .seg (true mass)
    sprintf(temp_filename, "%s.mor", in_filename);
    if ((fptr_mor = fopen(temp_filename,"rb")) == NULL) {
        sprintf(mor_filename, "%s.seg", in_filename);
        if ((fptr_mor = fopen(mor_filename,"rb")) == NULL) {
            printf("Segmented image of %s not found\n",in_filename);
            exit(0);
        }
    }
    else {
        sprintf(mor_filename, "%s.mor", in_filename);
        printf("Segmented image file is %s\n", mor_filename);
    }


    /*
    Extract ROI_Y and ROI_X from header of original image,
    ROI_Y(# of rows) is on the position #18 word
    ROI_X(# of columns) on #19 word
    */
    hn = fread(header, sizeof(unsigned short), 128, fptr);
    //Get file handle and file length
    fhandle = _fileno(fptr);
    ROI_size = _filelength(fhandle);

    //Check the format of the header --- ZUBO format
    if ((header[17] * header[18] + 256) != ROI_size) {
        ROI_X = ((header[17] & 0x00ff) << 8) | ((header[17] & 0xff00) >> 8);
        ROI_Y = ((header[18] & 0x00ff) << 8) | ((header[18] & 0xff00) >> 8);
    }
    else {
        ROI_X = header[17];
        ROI_Y = header[18];
    }

    //Check the format of the header --- TIFF format
/*  if ((header[16] * header[22] + 256) != ROI_size) {
        ROI_X = ((header[16] & 0x00ff) << 8) | ((header[16] & 0xff00) >> 8);
        ROI_Y = ((header[22] & 0x00ff) << 8) | ((header[22] & 0xff00) >> 8);
    }
    else {
        ROI_X = header[16];
        ROI_Y = header[22];
    }
*/

    //printf("The number of items read from header is %d \n", hn);
    printf("The size of the ROI image is %d %d \n", ROI_Y, ROI_X);


    ROI = (unsigned char *) malloc(sizeof(unsigned char) * ROI_X * ROI_Y);
    img_buffer = (unsigned char *) malloc(sizeof(unsigned char) * ROI_X);


    /*
    Read in original ROI image
    */
    for (i = 0; i < ROI_Y; i++) {
        fread(img_buffer, sizeof(unsigned char), ROI_X, fptr);
        for (j = 0;j < ROI_X;j++) {
            *(ROI + i * ROI_X + j) = *(img_buffer + j);
        }
    }

    /*
    Extract ROI_Y and ROI_X from header of segmented file,
    ROI_Y(# of rows) is on the position #18 word
    ROI_X(# of columns) on #19 word
    */
    hn_mor = fread(header_mor, sizeof(unsigned short), 128, fptr_mor);
    //Get file handle and file length
    fhandle_mor = _fileno(fptr_mor);
    ROI_size_mor = _filelength(fhandle_mor);

    //Check the format of the header
    if ((header_mor[17] * header_mor[18] + 256) != ROI_size_mor) {
        ROI_X_MOR = ((header_mor[17] & 0x00ff) << 8) | ((header_mor[17] & 0xff00) >> 8);
        ROI_Y_MOR = ((header_mor[18] & 0x00ff) << 8) | ((header_mor[18] & 0xff00) >> 8);
    }
    else {
        ROI_X_MOR = header_mor[17];
        ROI_Y_MOR = header_mor[18];
    }
```

```
//printf("The number of items read from header is %d \n", hn_mor);
printf("The size of the segmented ROI image is %d %d \n", ROI_Y_MOR, ROI_X_MOR);

if ((ROI_X != ROI_X_MOR) | (ROI_Y != ROI_Y_MOR)) {
    printf("Original and segmented files do NOT match!\n");
    exit(0);
}


ROI_MOR = (unsigned char *) malloc(sizeof(unsigned char) * ROI_X_MOR * ROI_Y_MOR);
img_buffer_mor = (unsigned char *) malloc(sizeof(unsigned char) * ROI_X_MOR);

/*
Read in segmented ROI image
*/
for (i = 0; i < ROI_Y_MOR; i++) {
    fread(img_buffer_mor, sizeof(unsigned char), ROI_X_MOR, fptr_mor);
    for (j = 0; j < ROI_X_MOR; j++) {
        *(ROI_MOR + i * ROI_X_MOR + j) = *(img_buffer_mor + j);
    }
}

fclose(fptr);
fclose(fptr_mor);



/* Scan image */

for (theda = 0; theda <= 135; theda += 45) {
    switch (theda) {
        case 0:
            a = 0;
            b = d;
            break;
        case 45:
            a = -d;
            b = d;
            break;
        case 90:
            a = -d;
            b = 0;
            break;
        case 135:
            a = -d;
            b = -d;
            break;
    }
    for (i = 0; i < ROI_Y; i++) {
        for (j = 0; j < ROI_X; j++) {
            /*
            Scan the image and search for any pixel pairs satisfied the
            the directional angle theda and distance d. The elements in
            SGLD matrix are the numbers of pixel pairs found.
            Total_pix_pair is the total number of pixel pairs.
            */
            temp_i = i + a;
            temp_j = j + b;
            if ((0 <= temp_i) && (temp_i < ROI_Y) && (0 <= temp_j) && (temp_j < ROI_X))
```

```
 {
                    gray_a = *(ROI + i * ROI_X + j); // gray level of pixel (i,j) in origin
al image
                    gray_b = *(ROI + temp_i * ROI_X + temp_j);
                    gray_a_mor = *(ROI_MOR + i * ROI_X_MOR + j);//gray level in segmented i
mage
                    gray_b_mor = *(ROI_MOR + temp_i * ROI_X_MOR + temp_j);
                    //Calculate the segmented area only
                    if ((gray_a_mor == 0) & (gray_b_mor == 0)) {
                    //gray_i and gray_j (range from 0 - 255) are the actual row and column
indices.
                    *(SGLD_ptr + GRAY_LEVEL * gray_a + gray_b) += 1;
                    total_pix_pair += 1;
                    }
                    //Calculate outside of segmented area only
                    if ((gray_a_mor != 0) & (gray_b_mor != 0)) {
                    //gray_i and gray_j (range from 0 - 255) are the actual row and column
indices.
                    *(SGLD_out_ptr + GRAY_LEVEL * gray_a + gray_b) += 1;
                    total_pix_pair_out += 1;
                    }

                }
            }
        }
    }
        /*
        Calculate the probabilities of each pixel pair found for segmented area.
        */
        for (i = 0; i < GRAY_LEVEL; i++) {
            for (j = 0; j < GRAY_LEVEL; j++) {
                *(SGLD_ptr + GRAY_LEVEL * i + j) /= total_pix_pair;
            }
        }
        /*
        Calculate the probabilities of each pixel pair found for outside of segmented area.
        */
        for (i = 0; i < GRAY_LEVEL; i++) {
            for (j = 0; j < GRAY_LEVEL; j++) {
                *(SGLD_out_ptr + GRAY_LEVEL * i + j) /= total_pix_pair_out;
            }
        }


}

void print_SGLD_matrix(int d, float *SGLD_ptr)
{
    int i, j;

    printf("\n%s %d\n", "Spatial Gray Level Dependence Matrix, with d = ", d);
    for (i = 0; i < GRAY_LEVEL; i++) {
        for (j = 0; j < GRAY_LEVEL; j++) {
            printf("%f", *(SGLD_ptr + GRAY_LEVEL * i + j));
        }
        printf("\n");
    }
}

void write_output_file(double energy_tf, double correlation_tf, double inertia_tf, double e
ntropy_tf,
    double inv_diff_moment_tf, double sum_average_tf, double sum_entropy_tf,double diff_ent
ropy_tf, double even_tf)
{
    char output_filename[20];
    FILE *output_fptr;

//  printf("Enter output filename\n");
//  gets(out_filename);

    strcpy(output_filename, "out.txt");

    if ((output_fptr = fopen(output_filename, "a")) == NULL) {
        printf("Cannot open output file %s \n",output_filename);
        exit(1);
    }


    /*fprintf(output_fptr, "\n%s\n", in_filename);
    fprintf(output_fptr, "%s\n", mor_filename);

    fprintf(output_fptr, "%s\n%s %f\n%s %f\n%s %f\n%s %f\n%s %f\n%s %f\n%s %f\n%s %f\n%s %f
\n",
            "Texture features computed:",
            "Energy = ", energy_tf,
            "Correlation =", correlation_tf,
            "Inertia = ", inertia_tf,
            "Entropy = ", entropy_tf,
            "Inverse difference moment = ", inv_diff_moment_tf,
            "Sum average = ", sum_average_tf,
            "Sum entropy = ", sum_entropy_tf,
            "Difference entropy = ", diff_entropy_tf,
            "Even = ", even_tf);*/
    fprintf(output_fptr, "%f\n%f\n%f\n%f\n%f\n%f\n%f\n%f\n%f\n",
            energy_tf, correlation_tf, inertia_tf, entropy_tf,
            inv_diff_moment_tf, sum_average_tf, sum_entropy_tf,diff_entropy_tf, even_tf);

    fclose(output_fptr);
}


void verify_SGLD_matrix(float *SGLD_ptr)
{
    int i, j, n = 1;

    //Extract non-zero elements in SGLD matirx
    printf("i, j and non-zero SGLD element \n");
    for (i = 0; i < GRAY_LEVEL; i++) {
        for (j = 0; j < GRAY_LEVEL; j++) {
            if (*(SGLD_ptr + GRAY_LEVEL * i + j) != 0) {
                printf("%d %d %d %f\n", n, i, j, *(SGLD_ptr + GRAY_LEVEL * i + j));
                n++;
            }
        }
    }
}
```

```
double energy(int d, float *SGLD_ptr)
{
    int i, j;
    double energy_feature = 0;

    /*
    Energy feature is the sum of square of each element of SGLD matrix.
    */

    for (i = 0; i < GRAY_LEVEL; i++) {
        for (j = 0; j < GRAY_LEVEL; j++) {
            energy_feature += pow(*(SGLD_ptr + GRAY_LEVEL * i + j), 2);
        }
    }
//  printf("\n%s %d %s %f\n", "The energy, with d = ", d, ", is ", energy_feature);

    return energy_feature;
}

double correlation(int d, float *SGLD_ptr)
{
    int i, j;
    /*
    mean_px is sum(i) of (i * sum(j) of probability_theda_d(i,j)).
    mean_py is sum(j) of (j * sum(i) of probability_theda_d(i,j)).
    variance_px is sum(i) of ((i - mean_px)^2 * sum(j) of probability_theda_d(i,j)).
    variance_py is sum(j) of ((j - mean_px)^2 * sum(i) of probability_theda_d(i,j)).
    */
    double mean_px = 0, variance_px = 0, mean_py = 0, variance_py = 0;
    /*
    marg_prob_i is sum(j) of probability_theda_d(i,j).
    marg_prob_j is sum(i) of probability_theda_d(i,j).
    */
    double marg_prob_i[GRAY_LEVEL] = {0}, marg_prob_j[GRAY_LEVEL] = {0};
    double correlation_feature = 0;

    for (i = 0; i < GRAY_LEVEL; i++) {
        for (j = 0; j < GRAY_LEVEL; j++){
            marg_prob_i[i] += *(SGLD_ptr + GRAY_LEVEL * i + j);
        }
        mean_px += i * marg_prob_i[i];
    }

    for (j = 0; j < GRAY_LEVEL; j++) {
        for (i = 0; i < GRAY_LEVEL; i++){
            marg_prob_j[j] += *(SGLD_ptr + GRAY_LEVEL * i + j);
        }
        mean_py += j * marg_prob_j[j];
    }

    for (i = 0; i < GRAY_LEVEL; i++) {
        variance_px += (i - mean_px) * (i - mean_px) * marg_prob_i[i];
    }

    for (j = 0; j < GRAY_LEVEL; j++) {
        variance_py += (j - mean_py) * (j - mean_py) * marg_prob_j[j];
    }

    /*
    Correlation feature calculation
    */
    for (i = 0; i < GRAY_LEVEL; i++) {
        for (j = 0; j < GRAY_LEVEL; j++) {
            correlation_feature += (i - mean_px) * (j - mean_py) * (*(SGLD_ptr + GRAY_LEVEL
 * i + j));
        }
    }
    correlation_feature = correlation_feature / sqrt(variance_px * variance_py);
//  printf("\n%s %d %s %f\n", "The correlation, with d = ", d, ", is ", correlation_feature
);

    return correlation_feature;

}

double inertia(int d, float *SGLD_ptr)
{
    int i, j;
    double inertia_feature = 0;

    for (i = 0; i < GRAY_LEVEL; i++) {
        for (j = 0; j < GRAY_LEVEL; j++) {
            inertia_feature += (i - j) * (i - j) * (*(SGLD_ptr + GRAY_LEVEL * i + j));
        }
    }
//  printf("\n%s %d %s %f\n", "The inertia, with d = ", d, ", is ", inertia_feature);
    return inertia_feature;

}

double entropy(int d, float *SGLD_ptr)
{
    int i, j;
    double entropy_feature = 0;
    float lgv;

    for (i = 0; i < GRAY_LEVEL; i++) {
        for (j = 0; j < GRAY_LEVEL; j++) {
            if (*(SGLD_ptr + GRAY_LEVEL * i + j) != 0){
            lgv = (float)(log(*(SGLD_ptr + GRAY_LEVEL * i + j)) / log(2.0));
            entropy_feature += (-1) * (*(SGLD_ptr + GRAY_LEVEL * i + j)) * lgv;
            }
        }
    }
    //printf("\n%s %d %s %f\n", "The entropy, with d =  ", d, ", is ", entropy_feature);
    return entropy_feature;

}

double inv_diff_moment(int d, float *SGLD_ptr)
{
    int i, j;
    double inv_diff_moment_feature = 0;

    for (i = 0; i < GRAY_LEVEL; i++) {
        for (j = 0; j < GRAY_LEVEL; j++) {
            inv_diff_moment_feature +=  *(SGLD_ptr + GRAY_LEVEL * i + j) / (1.0 + (i - j) *
 (i - j));
```

```
            }
        }
        //printf("\n%s %d %s %f\n", "The inverse difference moment, with d = ", d, ", is ", inv
_diff_moment_feature);
        return inv_diff_moment_feature;
}

double sum_average(int d, float *SGLD_ptr)
{
        int i, j, k;
        /*
        prob_k is sum of elements in SGLD matrix with i+j=k
        */
        double prob_k, sum_average_feature = 0;

        for (k = 0; k <= (2 * GRAY_LEVEL - 2); k++) {
            prob_k = 0;
            for (i = 0; i < GRAY_LEVEL; i++) {
                for (j = 0; j < GRAY_LEVEL; j++) {
                    if ((i + j) == k)
                        prob_k += *(SGLD_ptr + GRAY_LEVEL * i + j);
                }
            }
            sum_average_feature += k * prob_k;
        }
        //printf("\n%s %d %s %f\n", "The sum average, with d = ", d, ", is ", sum_average_featu
re);
        return sum_average_feature;
}

double sum_entropy(int d, float *SGLD_ptr)
{
        int i, j, k;
        double prob_k = 0, sum_entropy_feature = 0;

        for (k = 0; k < 2 * GRAY_LEVEL - 2; k++) {
            prob_k = 0;
            for (i = 0; i < GRAY_LEVEL; i++) {
                for (j = 0; j < GRAY_LEVEL; j++) {
                    if (i + j == k)
                        prob_k += *(SGLD_ptr + GRAY_LEVEL * i + j);
                }
            }
            if (prob_k != 0)
                sum_entropy_feature += (-1) * prob_k * (float)(log(prob_k) / log(2.0));
        }
        //printf("\n%s %d %s %f\n", "The sum entropy, with d = ", d, ", is ", sum_entropy_featu
re);
        return sum_entropy_feature;
}

double diff_entropy(int d, float *SGLD_ptr)
{
        int i, j, k;
        double prob_k = 0, diff_entropy_feature = 0;

        for (k = 0; k < GRAY_LEVEL - 2; k++) {
            prob_k = 0;
            for (i = 0; i < GRAY_LEVEL; i++) {
                for (j = 0; j < GRAY_LEVEL; j++) {
```

```
                if (abs(i - j) == k)
                    prob_k += *(SGLD_ptr + GRAY_LEVEL * i + j);
            }
        }
        if (prob_k != 0)
        diff_entropy_feature += -prob_k * (float)(log(prob_k) / log(2.0));
    }
    //printf("\n%s %d %s %f\n", "The difference entropy, with d = ", d, ", is ", diff_entro
py_feature);
    return diff_entropy_feature;
}

double even(int d, float *SGLD_ptr)
{
    int i, j;
    double even_feature = 0;

    for (i = 0; i < GRAY_LEVEL; i++) {
        for (j = 0; j < GRAY_LEVEL; j++) {
            if ((abs(i - j) == 0))
                even_feature += *(SGLD_ptr + GRAY_LEVEL * i + j);
        }
    }
    //printf("\n%s %d %s %f\n", "The even, with d = ", d, ", is ", even_feature);
    return even_feature;
}
```

```
/*  This program is designated to compute shape feature of segmented mass
    area.

    Zuyi Wang
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <memory.h>
#include <io.h>
#include <iostream.h>
#include "nr.h"
#include "nrutil.h"



char in_filename[100];
char output_filename[100];

void contour_fold(double *contour_X, double *contour_Y, int ROI_X, int ROI_Y,
                  int perimeter, double *contour_X_fold, double *contour_Y_fold,
                  int fold_line_point_num, double *a, double *b);
double contour_fit(double *a, double *b, double *contour_X_fold, double *contour_Y_fold,
                   int fold_line_point_num);


void main(int argc, char* argv[])
{
    if ( argc != 2 )
    {
        printf("No file in this directory!!!\n");
        exit(1);
    }

    int i, j, k, pix_in_seg = 0, perimeter = 0;
    int fhandle;
    int hn, ROI_X, ROI_Y;
    double *contour_X, *contour_Y, eccentricity;

    double ROI_size, compactness_sf;
    unsigned char *img_buffer, *ROI;
    unsigned short header[128];

    FILE *fptr;
    FILE *output_fptr;


    strcpy(in_filename, argv[1]);

    /*
    Open ROI file
    */
    if ((fptr = fopen(in_filename,"rb")) == NULL) {
        printf("Cannot open %s \n",in_filename);
        exit(0);
    }
    else {
        printf("Segmented image file is %s\n", in_filename);
    }


    hn = fread(header, sizeof(unsigned short), 128, fptr);

    //Check file handle and file length
    fhandle = _fileno(fptr);
    ROI_size = _filelength(fhandle);

    //Check the format of the header --- ZUBO format
    if ((header[17] * header[18] + 256) != ROI_size) {
        ROI_Y = ((header[17] & 0x00ff) << 8) | ((header[17] & 0xff00) >> 8);
        ROI_X = ((header[18] & 0x00ff) << 8) | ((header[18] & 0xff00) >> 8);
    }
    else {
        ROI_Y = header[17];
        ROI_X = header[18];
    }


    printf("The size of the ROI image is %d %d \n", ROI_Y, ROI_X);


    ROI = (unsigned char *) malloc(sizeof(unsigned char) * ROI_X * ROI_Y);
    img_buffer = (unsigned char *) malloc(sizeof(unsigned char) * ROI_X);



    //Read in original ROI image
    for (i = 0; i < ROI_Y; i++) {
        fread(img_buffer, sizeof(unsigned char), ROI_X, fptr);
        for (j = 0;j < ROI_X;j++) {
            *(ROI + i * ROI_X + j) = *(img_buffer + j);
        }
    }

    for (i = 0; i < ROI_Y; i++) {
        for (j = 0; j < ROI_X; j++) {
            if (*(ROI + i * ROI_X + j) == 0) {
                pix_in_seg ++;
                if ((*(ROI + (i - 1) * ROI_X + j) != 0) | (*(ROI + (i + 1) * ROI_X + j) !=
0) | (*(ROI + i * ROI_X + (j - 1)) != 0) | (*(ROI + i * ROI_X + (j + 1)) != 0))
                    perimeter ++;
            }
        }
    }


    contour_X = (int *) malloc(sizeof(int) * perimeter);
    contour_Y = (int *) malloc(sizeof(int) * perimeter);

    //Extract contour of segmented area
    k = 0;
    for (i = 0; i < ROI_Y; i++) {
        for (j = 0; j < ROI_X; j++) {
            if ((*(ROI + (i - 1) * ROI_X + j) != 0) | (*(ROI + (i + 1) * ROI_X + j) != 0) |
 (*(ROI + i * ROI_X + (j - 1)) != 0) | (*(ROI + i * ROI_X + (j + 1)) != 0)) {
                *(contour_Y + k) = i;
                *(contour_X + k) = j;
                k++;
            }
        }
    }


    //Name output file
    strcpy(output_filename, "out_shape.txt");

    if ((output_fptr = fopen(output_filename, "a")) == NULL) {
        printf("Cannot open output file %s \n",output_filename);
        exit(1);
    }


    //Fit straight line
    contour_fold(*contour_Y, *contour_X, ROI_X, ROI_Y, perimeter, *contour_Y_fold,
        *contour_X_fold, fold_line_point_num, *a, *b);

    //Fit curve
    contour_fit(*a, *b, *contour_X_fold, *contour_Y_fold, fold_line_point_num);

    //Print results
    printf("\n%s%f%s%f%s%f", "eccentricity=", eccentricity, "ry=", parameter[2], "rx=", par
ameter[3]);
    printf("\n%s%f", "chi square =", chisq);


}



// Arrange contour for curve fitting
void contour_fold(double *contour_Y, double *contour_X, int ROI_X, int ROI_Y, int perimeter
, double *contour_Y_fold, double *contour_X_fold, int fold_line_point_num, double *a, doubl
e *b)
{
    int i, j, k, n;
    int mwt = 0;
    double *fold_line_x, *fold_line_y, *fold_line_temp, fold_line_point = 0.0;
    double *sig, chi2, old_chi2, *q, *siga, *sigb;
    double cos_a, sin_a;
    double *fold_line_x_new, *fold_line_y_new;
    double contour_point_temp, *contour_Y_new, *contour_X_new;



    fold_line_x = (double *) malloc(sizeof(double) * ROI_Y);
    fold_line_y = (double *) malloc(sizeof(double) * ROI_Y);


    fold_line_point_num = 0;
    for (i = 0; i < ROI_Y; i++) {
        n = 0; // # of contour points with same y
        fold_line_point = 0;
        for (k = 0; k < perimeter; k++) {
            if(*(contour_Y + k) = i) {
                fold_line_point += *(contour_X + k);
                n++;
            }
        }
        if (fold_line_point != 0) {
            *(fold_line_x + fold_line_point_num)  = fold_line_point / n;
            fold_line_point_num++;
        }
    }

    for (i = 0; i < ROI_Y; i++) {
        if(*(fold_line_temp + i) != 0) {
            *(fold_line_x + fold_line_point_num) = *(fold_line_temp + i);
            *(fold_line_y + fold_line_point_num) = i;
            fold_line_point_num++;
        }
    }

    sig = (double *) malloc(sizeof(double) * fold_line_point_num);
    contour_X_fold = (double *) malloc(sizeof(double) * fold_line_point_num);


    chi2 = 0;
    do {

        old_chi2 = chi2;
        fit(fold_line_y, fold_line_x, fold_line_point_num, sig, mwt, a, b, siga, sigb, &chi
2, q);

    }
    while(abs(*chi2 - old_chi2) < 0.1);

    printf("\n%s%f %s%f %s%f\n", "a=", *a, "b=", *b, "chi2=", chi2);

    /*calculate cos(angle) and sin(angle) from b (tan(angle))
    cos=1/(sqrt(tan^2+1)), sin=tan/(sqrt(tan^2+1))
    */

    cos_a = 1 / (sqrt((*b) * (*b) + 1));
    sin_a = (*b) / (sqrt((*b) * (*b) + 1));

    /*coordinate rotation
    y_new = y*cos_a+x*sin_a
    x_new = x*cos_a-y*sin_a
    */

    for (i = 0; i < perimeter; i++) {
        *(contour_Y_new + i) = *(contour_Y + i) * cos_a + *(contour_X + i) * sin_a;
        *(contour_X_new + i) = *(contour_X + i) * cos_a - *(contour_Y + i) * sin_a;
    }


    for (i = 0; i < fold_line_point_num; i++) {
        *(contour_Y_fold + i) = *(fold_line_y + i) * cos_a + *(fold_line_x + i) * sin_a;
    }



    /*Calculate distances from the points on the contour
    on both sides of the fold line
    */
    for (i = 0; i < fold_line_point_num; i++) {
        contour_point_temp = *(contour_Y_new + i);
        k = 0;
        *(contour_X_fold + i)= *(contour_X_new + i);
        for (j = 0; j < perimeter; j++) {
            if (*(contour_Y_new + j) == contour_point_temp) {
                 *(contour_X_fold + i) += *(contour_X_new + j);
                 k++;
            }
            *(contour_X_fold + i) /= (k + 1);
        }
    }


}

//Define function, and derivations of each parameter
void f_ellipse(double y, double parameter[], double *x, double dx_dparameter[], int num_par
ameter)
{
    double arg;

    arg = sqrt(1 - (y - parameter[0]) * (y - parameter[0]) / (parameter[2] * parameter[2]))
;

    dx_dparameter[0] = parameter[3] * arg * (y - parameter[0]) / (parameter[2] * parameter[
2]);
    dx_dparameter[1] = 1;
    dx_dparameter[2] = parameter[3] *arg * (y - parameter[0]) * (y - parameter[0]) / (param
eter[2] * parameter[2] *parameter[2]);
    dx_dparameter[3] = arg;

}


double contour_fit(double *a, double *b, double *contour_X_fold, double *contour_Y_fold, in
t fold_line_point_num)
{

    int i, j, k, *iparameter, iter;
    long num_parameter = 4;

    double center_x, center_y; //center of ellipse
    double ry, rx, eccentricity;
    double parameter[4], *sig, chisq, old_chisq, alamda, **covar, **alpha;

    center_x = *a;
    center_y = (*(contour_Y_fold) + *(contour_Y_fold + fold_line_point_num - 1)) / 2;

    ry = (*(contour_Y_fold + fold_line_point_num - 1) - *(contour_Y_fold)) / 2;
    rx = *(conyour_X_fold + (fold_line_point_num + 1) / 2);

    iparameter = ivector(1, num_parameter);
    sig = vector(1, fold_line_point_num);
    covar = matrix(1, num_parameter, 1, num_parameter);
    alpha = matrix(1, num_parameter, 1, num_parameter);

    //Initialize iparameters to 1 for all parameter to be fitted
    for (i=1; i <= num_parameter; i++) iparameter[i] = 1;

    //Initialize parameters
    parameter[0] = center_y;
    parameter[1] = center_x;
    parameter[2] = ry;
    parameter[3] = rx;

    alamda = -1;
    chisq = 0;

    do  {
        old_chisq = chisq;
        mrqmin(contour_Y_fold, contour_X_fold, sig, fold_line_point_num, parameter, iparame
ter,
            num_parameter, covar, alpha, &chisq, f_ellipse, &alamda);

    }
    while (abs(old_chisq - chisq) < 0.1);

    alamda =0;
    mrqmin(contour_Y_fold, contour_X_fold, sig, fold_line_point_num, parameter, iparameter,
            num_parameter, covar, alpha, &chisq, f_ellipse, &alamda);

    if (parameter[2] >= parameter[3]) {
        eccentricity = sqrt(1 - (parameter[3] / parameter[2]) * (parameter[3] / parameter[2
]));
        }
    else {
        eccentricity = sqrt(1 - (parameter[2] / parameter[3]) * (parameter[2] / parameter[3
]));
        }

}
```

```
/* note #undef's at end of file */
#define NRANSI
#include "nrutil.h"

void mrqmin(double x[], double y[], double sig[], int ndata, double a[], int ia[],
    int ma, double **covar, double **alpha, double *chisq,
    void (*funcs)(double, double [], double *, double [], int), double *alamda)
{
    void covsrt(double **covar, int ma, int ia[], int mfit);
    void gaussj(double **a, int n, double **b, int m);
    void mrqcof(double x[], double y[], double sig[], int ndata, double a[],
        int ia[], int ma, double **alpha, double beta[], double *chisq,
        void (*funcs)(double, double [], double *, double [], int));
    int j,k,l;
    static int mfit;
    static double ochisq,*atry,*beta,*da,**oneda;

    if (*alamda < 0.0) {
        atry=vector(1,ma);
        beta=vector(1,ma);
        da=vector(1,ma);
        for (mfit=0,j=1;j<=ma;j++)
            if (ia[j]) mfit++;
        oneda=matrix(1,mfit,1,1);
        *alamda=0.001;
        mrqcof(x,y,sig,ndata,a,ia,ma,alpha,beta,chisq,funcs);
        ochisq=(*chisq);
        for (j=1;j<=ma;j++) atry[j]=a[j];
    }
    for (j=1;j<=mfit;j++) {
        for (k=1;k<=mfit;k++) covar[j][k]=alpha[j][k];
        covar[j][j]=alpha[j][j]*(1.0+(*alamda));
        oneda[j][1]=beta[j];
    }
    gaussj(covar,mfit,oneda,1);
    for (j=1;j<=mfit;j++) da[j]=oneda[j][1];
    if (*alamda == 0.0) {
        covsrt(covar,ma,ia,mfit);
        covsrt(alpha,ma,ia,mfit);
        free_matrix(oneda,1,mfit,1,1);
        free_vector(da,1,ma);
        free_vector(beta,1,ma);
        free_vector(atry,1,ma);
        return;
    }
    for (j=0,l=1;l<=ma;l++)
        if (ia[l]) atry[l]=a[l]+da[++j];
    mrqcof(x,y,sig,ndata,atry,ia,ma,covar,da,chisq,funcs);
    if (*chisq < ochisq) {
        *alamda *= 0.1;
        ochisq=(*chisq);
        for (j=1;j<=mfit;j++) {
            for (k=1;k<=mfit;k++) alpha[j][k]=covar[j][k];
            beta[j]=da[j];
        }
        for (l=1;l<=ma;l++) a[l]=atry[l];
    } else {
        *alamda *= 10.0;
        *chisq=ochisq;
    }
}
#undef NRANSI
```

```
#include <math.h>
#define NRANSI
#include "nrutil.h"

void fit(double x[], double y[], int ndata, double sig[], int mwt, double *a,
	double *b, double *siga, double *sigb, double *chi2, double *q)
{
	double gammq(double a, double x);
	int i;
	double wt,t,sxoss,sx=0.0,sy=0.0,st2=0.0,ss,sigdat;

	*b=0.0;
	if (mwt) {
		ss=0.0;
		for (i=1;i<=ndata;i++) {
			wt=1.0/SQR(sig[i]);
			ss += wt;
			sx += x[i]*wt;
			sy += y[i]*wt;
		}
	} else {
		for (i=1;i<=ndata;i++) {
			sx += x[i];
			sy += y[i];
		}
		ss=ndata;
	}
	sxoss=sx/ss;
	if (mwt) {
		for (i=1;i<=ndata;i++) {
			t=(x[i]-sxoss)/sig[i];
			st2 += t*t;
			*b += t*y[i]/sig[i];
		}
	} else {
		for (i=1;i<=ndata;i++) {
			t=x[i]-sxoss;
			st2 += t*t;
			*b += t*y[i];
		}
	}
	*b /= st2;
	*a=(sy-sx*(*b))/ss;
	*siga=sqrt((1.0+sx*sx/(ss*st2))/ss);
	*sigb=sqrt(1.0/st2);
	*chi2=0.0;
	*q=1.0;
	if (mwt == 0) {
		for (i=1;i<=ndata;i++)
			*chi2 += SQR(y[i]-(*a)-(*b)*x[i]);
		sigdat=sqrt((*chi2)/(ndata-2));
		*siga *= sigdat;
		*sigb *= sigdat;
	} else {
		for (i=1;i<=ndata;i++)
			*chi2 += SQR((y[i]-(*a)-(*b)*x[i])/sig[i]);
		if (ndata>2) *q=gammq(0.5*(ndata-2),0.5*(*chi2));
	}
}
#undef NRANSI
```

```
% VE_RUN Hierarchical interactive data visualization
%
%     VE_RUN(DATA, VTYPE, itr, LABELS)
%
%  DATA      N x d data matrix
%
%  VTYPE(VL) The visualisation type:
%             X:
%              0 = Plain
%              1 = with LABELS
%              2 = no labels, but `depth colouring'
%            01X = X is as 0-2, but with a white background
%            1XX =XX is as 1X, but with REP # displayed
%
%  LABELS    N x 1 vector of integer class labels
%            (optional). Note that these are never used
%            in determining the model, but can be useful
%            when included in the plots.
%
% See also: VE_SHOW
%

function ve_run(D,vl,itr,labels)
% Declare global variables for storing the hierarchical tree structure.
%
global Structure;
global SaveW;
global SaveP;
global SavePROP;
global SaveMU;
global vl1;
global vl2;
global vl3;
global D_rep;
global finishedAll;

finishedAll=0;
% Parameters
CENTRE_FSIZE = 14;
vl1=vl;
if vl1>99
   vl3=1;
   vl1=vl1-100;
else
   vl3=0;
end
if vl1>9
   vl2=1;
   vl1=vl1-10;
else
   vl2=0;
end

% Set default black background
if vl2==1
 whitebg(0,'w');
else
 whitebg(0,'k');
end

if nargin<3 | nargin>4
   error('ve_run requires 3 (or optionally 4) arguments.')
end

% Fix the latent dimensionality at 2 for visualization
%
q=2;
```

```
Structure = [0 0];

[n p]=size(D);
if nargin==2
   labels=[];
   if rem(vl,2)
     fprintf('! Warning: VLevel %d requested, but no labels supplied.\n',vl)
     vl1 = vl1-1;
     vl=vl-1;
     fprintf('! Attention:Using VLevel %d instead.\n',vl)
   end
else
   [nlabel plabel]=size(labels);
   if nlabel ~= n
     fprintf('! Warning: row# of labels is different from Data.\n')
     vl1 = vl1-1;
     vl=vl-1;
     fprintf('! Attention:Using VLevel %d instead.\n',vl)
   end
end
D_rep=zeros(n,1);  % corresponding to D, rep # stored in it
if vl3==1
   D_rep=labels;
   if vl1==1
    ll1= labels(1);
    ll2= 0;
    for i = 1:n
      if labels(i)~=ll1
           ll2=1;
           ll1=labels(i);
      else
           ll2=ll2+1;
      end
      D_rep(i)=ll2;
    end
   else
    for i = 1:n
       D_rep(i)=i;
    end
   end
end

% Centre the data
%
mu = mean(D);    % first dimension
D  = D-ones(n,1)*mu;  % the mean should be zero after transform
%
% Close windows to clear the screen, but leave figure(1) intact
%
figs = get(0,'Children');
close(figs(figs~=1))

% Top level latent variable projection
% W:sorted top 2 eigen vectors, U:sorted total eigen vector matrix
% P:averaged eigen value from q+1 to p.    [n p]=size(D). q:2
[W P U] = ve_zero(D,n,p,q);

% Plot the projection
% fig2-MDL, fig1-reserved for center selection of all level
fig=3;
%function ve_mvis(level,D,W,MU,R,n,p,q,labels,Zjk)
%function ve_vis_t(D,W,MU,R,n,p,q,LABELS,Zjk,BLOB_SIZE)
%MU:center selected, R:UK, D:n X p matric original data, q:2
figure(fig)
clf
ve_mvis(1,D,W,zeros(1,p),ones(n,1),n,p,q,labels,ones(n,1));
title('Top Level Visualization')
set(gcf,'MenuBar','None')
```

```
hold on
%
%function [ k0, MU_t, w0  ] =
%   ve_mselect(level, fig, D, W, MU, R, n, p, q, labels, Zjk);
[k0,MU_t,w0,Zjk_t]=ve_mselect(1,fig,D,W,zeros(1,p),ones(n,1),n,p,q,labels,ones(n,1));

% The latent cluster centres have been selected, so transform them into the
% data space, and generate the mixture model.
%
fprintf('\n! OK. Generating Top Level Mixture Model.\n');
if finishedAll
   return;
end

fig=fig+1;
level=2;
vk0=ones(1,k0); % every point selected is belong to picture 1(top level)
% function newfig = ve_sub_new(uc,fig,level,D,MU,q,labels,w0,Zjk_t)
ve_sub_new(vk0,fig,level,D,MU_t,q,labels,w0,ones(n,1),itr);
%
% Finally, save all the parameters in the requisite file.
%
%if SaveFlag
%  eval(['save ' file_ ' SaveW SaveP SavePROP SaveMU Structure']);
%end
```

```
%model select
%MDL = -log(Lml) + 0.5*Ka*logN
%Ka = 3*K - 1;
%K0 is the optimal number of clusters
%X is the input data
%MU is centers that user selected in upper level
%Zk is the k column of Zjk of upper cluster
%W is the Wk of each cluster in upper level
%D, n x p, is p Dimention data of upper level
%fig, number of figure
%k1, k2, range of the number of clusters
%KK, is the number of clusters of upper level
%mselect.m

function [k0,MU_t,w0,Zjk] = ve_mselect(level, fig, D, W, MU, R, n, p, q, labels,Zk);
%[ N p ] = size( X );
global Structure;
global SaveW;
global SaveP;
global SavePROP;
global SaveMU;
global vl1;
global vl2;
global vl3;
global D_rep;
global finishedAll;

CENTRE_FSIZE = 10;
figure (fig);
% Obtain button presses
%
fprintf('---------------------------------------\n');
fprintf('| Left Click on some center points now!\n')
fprintf('| [Right Click terminates...]\n');
fprintf('| Q          = Quit all remaining plots\n');
fprintf('---------------------------------------\n');
finished = 0;
NEWMU = [];
points = 0;
while ~finished
  [x1 x2 button] = ginput(1);
  if button == 'q' | button == 'Q'
      finishedAll=1;
      return;
  end
  if (button==3 | button==2) & points>1
    finished = 1;
  elseif button == 1
    NEWMU   = [NEWMU; x1 x2];
    points  = size(NEWMU,1);
    plot(x1,x2,'k.','MarkerSize',CENTRE_FSIZE*3-5);
    text(x1,x2,num2str(points),'FontWeight','Bold',...
      'FontSize',CENTRE_FSIZE-3,'Color','g','HorizontalAlignment','center');
  end
end
hold on;

%[n p] = size(D);
[k0 p1] = size(NEWMU);

tpi=pi*2;  %2 PI

X =round(W'*D'); %X:x space data - 2D

% Create the histogram of training data
low_x=min(X(1,:));
low_y=min(X(2,:));
high_x=max(X(1,:));
```

```
high_y=max(X(2,:));

begin_x=1;
begin_y=1;

end_x=high_x-low_x+1;
end_y=high_y-low_y+1;

rk_x=1:1:end_x;
rk_y=1:1:end_y;

nrk_x=zeros(1,end_x);
nrk_y=zeros(1,end_y);
nrk_xy=zeros(end_x,end_y);

for j=1:n,
   nrk_x(1,X(1,j)-low_x+1)=nrk_x(1,X(1,j)-low_x+1)+1;
   nrk_y(1,X(2,j)-low_y+1)=nrk_y(1,X(2,j)-low_y+1)+1;

   nrk_xy(X(1,j)-low_x+1,X(2,j)-low_y+1)=nrk_xy(X(1,j)-low_x+1,X(2,j)-low_y+1)+1;
end

hist_x=nrk_x/n;
hist_y=nrk_y/n;
hist_xy=nrk_xy/n;

%fprintf('----------------------------------------\n')
fprintf('! Top Level window culculation begins ...\n');
%Use 2D EM algorithm to perform the local training.
%error=1.0/n/n;
for i = 1:k0
  w0(i)=1./k0;
  Var0(:,2*i-1:2*i) = 120*eye(2);
end

error=0.0000001;
 %error=0.000001;
err=100;
err1=err;
MUK=NEWMU;

sZk = sum(Zk);
while err > error,
      Fx = zeros(n,1);
      Gxn = zeros(n,k0);
      for kk = 1:k0
         for j = 1:n
           Gxn(j,kk) = sqrt(1./tpi)*exp( -(X(:,j)'-MUK(kk,:))*inv(Var0(:,2*kk-1:2*kk))...
               *(X(:,j)'-MUK(kk,:))'/2)/sqrt(det(Var0(:,2*kk-1:2*kk)));
         end
         Fx = Fx + w0(kk)*Gxn(:,kk);
      end
      MU1=[];
      for k = 1:k0
         Zjk(:,k) = w0(k)*Zk.*Gxn(:,k)./Fx;
         zz = sum(Zjk(:,k));
         w1(k) = zz/sZk;
         MU1(k,1) = sum(Zjk(:,k).*X(1,:)')/zz;
         MU1(k,2) = sum(Zjk(:,k).*X(2,:)')/zz;
         Ck = zeros(2,2);
         for j = 1:n
            Ck = Ck + Zjk(j,k)*(X(:,j)'-MUK(k,:))'*(X(:,j)'-MUK(k,:));
         end
         Var1(:,2*k-1:2*k) = Ck./zz;
      end
      w0 = w1;
      MUK = MU1;
      Var0 = Var1;
      for k=1:k0,
          c(k)=1/(tpi*sqrt(Var0(1,2*k-1)*Var0(2,2*k)));
      end

      p_xy=zeros(end_x,end_y);
      px=zeros(end_x,end_y);
      py=zeros(end_x,end_y);
      for k=1:k0,
        vx=2*Var0(1,2*k-1);
        vy=2*Var0(2,2*k);
        w0c=w0(k)*c(k);
        for ii=1:end_x,
           px(ii,:)=-(ii+low_x-1-MUK(k,1)).^2/vx;
        end
        for jj=1:end_y,
           py(:,jj)=-(jj+low_y-1-MUK(k,2)).^2/vy;
        end
        p_xy=p_xy+w0c*exp(px+py);
      end
      err2=sum(sum((p_xy-hist_xy).^2))/(end_x*end_y);
      err=abs(err2-err1);
      err1=err2;
      fprintf('*')
end  % while err>error

%MU2 = zeros(k0,2);
%index pointer to NEWMU
MU2=NEWMU;
%transform the means of X projection to the T data space
MU_t = (W*MU2')';
```

```
% VE_SUB Generate hierarchical submodel
%
function newfig = ve_sub(fig,level,D,MU,q,its,R,vl,labels)

% Global variables for saving the structure
%
global Structure;
global SaveW;
global SaveP;
global SavePROP;
global SaveMU;
global SaveFlag;
global vl1;
global vl2;
global vl3;
global D_rep;
global finishedAll;

% Parameters
% .
CENTRE_FSIZE = 14;
TITLE_FSIZE  = 14;

term = 0;

MON  = 5;

[n p] = size(D);
[m p] = size(MU);

[DR NR RR] = ve_reduc(D,n,p,R);
[W P PROP] = ve_inits(DR,MU,NR,m,p,q,RR); )    %lomub ] ]

newfig = fig+1;

finishedTraining = 0;
eTotal           = [];
while ~finishedTraining  %w1
  figure(1)
  set(1,'Name','Optimisation Monitor')
  set(1,'MenuBar','None')
  clf

  finishedTraining = 1;

  figure(fig)

  [x1 x2 button] = ginput(1);
  if button == 32
    finishedTraining = 0;
    fprintf('** OK **  Continuing optimisation ...\n');
  else
    finishedTraining = 1;
  end
end %w1

hold on
set(fig,'NumberTitle','Off')
fprintf('\n');

finishedAll = 0;
j = 0;
while j<m & ~finishedAll %w2
  j = j+1;
  term       = 1;
  NEWMU      = [];
  handles    = [];
  finishedSingle = 0;
```

```
subplot(suby,subx,j)
set(fig,'Name',[num2str(fig) '   *Selection Mode*   active plot: ' num2str(j) ])
ve_tick('on')
set(get(gca,'Title'),'Color','y')
fprintf('[Figure %d] Plot %d OPTIONS:\n',fig,j);
fprintf('\t Left Button   = Select centre in active plot\n')
fprintf('\t Right Button  = Optimise/skip active plot\n')
fprintf('\t R             = Reset centres\n')
fprintf('\t Q             = Quit all remaining plots\n')
fprintf('\t Numeric Key   = Change visualisation type\n')
while ~finishedSingle & ~finishedAll % w3

  [x1 x2 button] = ginput(1);

  if button >=48 & button <=57
    vl = button-48;
    if rem(vl,4)==1 & isempty(labels)
      vl = vl-1;
      fprintf(['Cannot change to visualisation level %d' ...
     ' - no class labels\n'], vl+1);
    else
      fprintf('Changing to visualisation level %d\n', vl);
      NEWMU   = [];
      handles    = [];
      clf
      ve_mvis(vl,D,W,P,MU,pjy,n,m,p,q,labels);
    end
    subplot(suby,subx,j)
    ve_tick('on')
    set(get(gca,'Title'),'Color','y')

  elseif (button == 2 | button == 3) & size(NEWMU,1) ~= 1
    finishedSingle = 1;

  elseif button == 'q' | button == 'Q'
    finishedAll = 1;

  elseif button == 'r' | button == 'R'
    delete(handles);
    NEWMU   = [];
    handles    = [];

  elseif button == 1
    NEWMU = [NEWMU; x1 x2];
    points    = size(NEWMU,1);
    handles = [handles ;plot(x1,x2,'b.','MarkerSize',CENTRE_FSIZE*4)];
    handles = [handles ;text(x1,x2,num2str(points),'FontWeight','Bold',...
          'FontSize',CENTRE_FSIZE,'Color','w','HorizontalAlignment','center')];
  end
end %w3

if finishedAll
 return;
end

nmu = size(NEWMU,1);
if nmu~=0
  NEWMU = NEWMU*W((j-1)*p+1:j*p,:)'+ ones(nmu,1)*MU(j,:);
  fprintf('\n** OK **  Generating Mixtures at level %d ...\n', fig-1);
  newfig = ve_sub(newfig,level+1,D,NEWMU,q,its,pjy(:,j),vl,labels);
  term = 0;
  figure(fig)
  Structure  = [Structure ; level term];
  if SaveFlag
    SaveW     = [SaveW ; W((j-1)*p+1:j*p,:)];
    SaveP     = [SaveP ; P(j)];
    SavePROP = [SavePROP; pjy(:,j)'];
    SaveMU    = [SaveMU; MU(j,:)];
```

```
      end
    elseif finishedSingle
      Structure  = [Structure ; level term];
      if SaveFlag
        SaveW    = [SaveW ; W((j-1)*p+1:j*p,:)];
        SaveP    = [SaveP ; P(j)];
        SavePROP = [SavePROP; pjy(:,j)'];
        SaveMU   = [SaveMU; MU(j,:)];
      end
    end

    ve_tick('off')
    set(get(gca,'Title'),'Color','w')
  end % w2

  if finishedAll
    for jj = j:m
      Structure   = [Structure ; level 1];
      if SaveFlag
        SaveW = [SaveW ; W((jj-1)*p+1:jj*p,:)];
        SaveP = [SaveP ; P(jj)];
        SavePROP = [SavePROP; pjy(:,jj)'];
        SaveMU   = [SaveMU; MU(jj,:)];
      end
    end
  end

  set(fig,'Name',[num2str(fig) ': Finalised'])
```

```
% PV_MVIS   Generate multiple visualisation plots at a single level
%
% uc: upper level cluster number

function ve_mvis(level,D,WW,MU,R,n,p,q,labels,Zjk)
global Structure;
global SaveW;
global SaveP;
global SavePROP;
global SaveMU;
global vl1;
global vl2;
global vl3;
global D_rep;
global finishedAll;

if level==1
   ve_vis_t(D,WW,MU,R,n,p,q,labels,Zjk);
   return;
end

[ulr ulc]=size(MU);
placey = 1;
placex = ulr;
if placex>=4 & placex<=6
  placex = ceil(placex/2);placey=2;
elseif placex>6 & placex<=12
  placex = ceil(placex/3);placey=3;
elseif placex>12
  placex = ceil(placex/4);placey=4;
end

scale = 5;
ve_fsize(placex*scale,placey*scale);
if vl2==1
  whitebg(gcf,'w');
else
  whitebg(gcf,'k');
end

for j = 1:ulr
  subplot(placey,placex,j)
   %Xm = D - ones(n,1)*MU(j,:)';
   %Xk = Xm.*Zjk(:,j);
   %[PCS, A, B, New] = princomp( Xk );
   %WW(:,2*j-1: 2*j) = PCS(:,1:2);
  ve_vis_t(D,WW(:,2*j-1: 2*j),MU(j,:),R,n,p,q,labels,Zjk(:,j));
  axis('square');
  ve_tick('off')
  set(gca,'Box','On')
  t_ = sprintf('picture %d ', j);
  title(t_, 'FontSize',12);

  %discardFraction = 100*(p-q)*P(j)/(p*P(j) + ...
  %   trace(W((j-1)*p+1:j*p,:)'*W((j-1)*p+1:j*p,:)));
  %t_ = sprintf('{\\bf %d} (%2.1f%%)', j, discardFraction);
  %title(t_, 'FontSize',12);
end
```

```
function [mean1_x,Zjk]=ve_sub_em(uc,D,mean0_x,Zk,w0,itr)
%[ucr ucc]=size(uc);
[k0 kc]=size(mean0_x);
[n p]=size(D);
fprintf('! EM start ....');
% Initializing the parameters of mixture of Gaussians
% Initinalize the covariance matrix

 %Use EM algorithm to perform the local training.
Var0=[];
for i = 1:k0
   Var0(1:p,(i-1)*p+1:i*p) = 20*eye(p);
end
tpi=2*pi;
MUK=mean0_x;
sZk = sum(Zk);
for itra=1:itr
%   while err > error,
    Fx = zeros(n,1);
    Gxn = zeros(n,k0);
    for kk = 1:k0
       for j = 1:n
         Gxn(j,kk) = sqrt(1./tpi)*exp(-(D(j,:)-MUK(kk,:))*inv(Var0(1:p,p*(kk-1)+
1:p*kk))...
            *(D(j,:)-MUK(kk,:))'/2)/sqrt(det(Var0(:,p*(kk-1)+1:p*kk)));
       end
       Fx = Fx + w0(kk)*Gxn(:,kk);
    end
    MU1=[];
    for k = 1:k0
      % Zjk(:,k) = w0(k)*Zk(:,uc(k)).*Gxn(:,k)./Fx;
      Zjk(:,k) = w0(k)*Gxn(:,k)./Fx;
      zz = sum(Zjk(:,k));
      %w1(k) = zz/sZk(uc(k));
      w1(k)=zz/n;
      for j=1:p
         MU1(k,j) = sum(Zjk(:,k).*D(:,j))/zz;
      end
      Ck = zeros(p,p);
      for j = 1:n
         Ck = Ck + Zjk(j,k)*(D(j,:)-MUK(k,:))'*(D(j,:)-MUK(k,:));
      end
      Var1(:,p*(k-1)+1:p*k) = Ck./zz;
    end
    w0 = w1;
    MUK = MU1;
    Var0 = Var1;
    %fprintf('*');
end  % while err>error
mean1_x=MUK;
for k = 1:k0
   Zjk(:,k) = Zjk(:,k).*Zk(:,uc(k));
end

fprintf(' end.\n');
```

```
% VE_VIS Plot a single visualization projection
%
% D: data, W:top 2 eigen vectors, MU: center selected for suitable dispaly
% R: for deep level display
function ve_vis_t(D,W,MU,R,n,p,q,LABELS,Zjk,BLOB_SIZE)
%
% Some useful definitions
%
% Font definitions for VTYPE 9
%
global Structure;
global SaveW;
global SaveP;
global SavePROP;
global SaveMU;
global vl1;
global vl2;
global vl3;
global D_rep;
global finishedAll;

FONT_NAME    = 'ZapfDingbats';
FONT_SYMBOLS    = [113 117 109 98 102 166 108];
FONT_SIZE    = 6;

% Size of font for box labels
CENTRE_FSIZE    = 8;

% Marker type for classes > 5
OVER5_MARKER_   = 'd';

% The following defines the number of colour gradations. Could increase this
% for 32k or 16M colour screens
COL_STEP = 16;

if nargin == 9
   BLOB_SIZE = 30;
end

if vl3 == 0
   BLOB_SIZE = 14;
end

R = R/max(max(R));

load cols.def
cols  = cols/255;
colsw = ones(5,3)-cols*0.75;
colsw(5,:) = [0.75 0.75 0.75];
symv = FONT_SYMBOLS;
sym  = setstr(symv);

THRESHOLD = 1/(2*COL_STEP);

X = (D-ones(n,1)*MU).*(ones(p,1)*Zjk')' * W ;

if (vl1 == 2) & BLOB_SIZE>0
  [U S V] = svd(W,0);
  distFactor = sqrt(sum(((D-ones(n,1)*MU)*(eye(p)-U*U'))'.^2)');
  distFactor = distFactor/max(distFactor(R>THRESHOLD));
  % These next lines should help with some color problems
  distFactor   = fix(distFactor * COL_STEP)/COL_STEP;
end
R  = fix(R*COL_STEP)/COL_STEP;

hold on

if BLOB_SIZE>0
```

```
[y index] = sort(R);
first = min(find(y>THRESHOLD));
for i = first:n
  lb=D_rep(index(i));
  if vl1~=9
    h = plot(X(index(i),1),X(index(i),2),'.');
    if vl1 == 0 | vl1 == 4
       set(h,'Color',[1 0 0]*R(index(i)),'MarkerSize',BLOB_SIZE)
       if vl3==1
          set(h,'MarkerSize',BLOB_SIZE*2-18)
          text(X(index(i),1),X(index(i),2),num2str(lb),'FontWeight','Bold',...
             'FontSize',CENTRE_FSIZE+2,'Color','y','HorizontalAlignment','center');
       end
    end
    if vl1 == 1
      if LABELS(index(i))>5
       set(h,'Marker',OVER5_MARKER_);
       set(h,'Color',R(index(i))*cols(LABELS(index(i))-5,:),...
     'MarkerSize',BLOB_SIZE,'LineWidth',1)
      else
       set(h,'Color',R(index(i))*cols(LABELS(index(i)),:),'MarkerSize',BLOB_SIZE)
      end
      if vl3==1
         set(h,'MarkerSize',BLOB_SIZE*2-18)
         text(X(index(i),1),X(index(i),2),num2str(lb),'FontWeight','Bold',...
             'FontSize',CENTRE_FSIZE+2,'Color','y','HorizontalAlignment','center');
      end
    end
    if vl1 == 2
      col = ([1 1 0] - distFactor(index(i))*[1 1 -1])*R(index(i));
      set(h,'Color',col,'MarkerSize',BLOB_SIZE)
      if vl3==1
         set(h,'MarkerSize',BLOB_SIZE*2-18)
         text(X(index(i),1),X(index(i),2),num2str(lb),'FontWeight','Bold',...
             'FontSize',CENTRE_FSIZE+2,'Color','y','HorizontalAlignment','center');
      end
    end

    if vl1 == 5
       if LABELS(index(i))>5
        set(h,'Marker',OVER5_MARKER_);
        set(h,'Color',[1 1 1] - R(index(i))*colsw(LABELS(index(i))-5,:),...
     'MarkerSize',BLOB_SIZE,'LineWidth',1)
       else
        set(h,'Color',[1 1 1] - R(index(i))*colsw(LABELS(index(i)),:),...
     'MarkerSize',BLOB_SIZE)
       end
       if vl3==1
          set(h,'MarkerSize',BLOB_SIZE*2-18)
          text(X(index(i),1),X(index(i),2),num2str(lb),'FontWeight','Bold',...
             'FontSize',CENTRE_FSIZE+2,'Color','y','HorizontalAlignment','center');
       end
    end
  else % VL1=9
    text(X(index(i),1),X(index(i),2),setstr(sym(LABELS(index(i)))),...
 'FontName', FONT_NAME,'FontSize', FONT_SIZE,...
 'Color',[1 1 1] - R(index(i)),'HorizontalAlignment','center');
  end
end   % for loop
limits = [min(X(index(first:n),1)) max(X(index(first:n),1))];
adjust = (limits(2)-limits(1))*0.025;
if adjust
  limits = limits + adjust*[-1 1];
  set(gca,'xlim',limits)
end
limits = [min(X(index(first:n),2)) max(X(index(first:n),2))];
adjust = (limits(2)-limits(1))*0.025;
if adjust
```

```
    limits = limits + adjust*[-1 1];
    set(gca,'ylim',limits)
  end

else      %BLOB_SIZE < 0
  %
  % Nasty hack to enable the plotting of boxes
  %
  PERBOX = 6;
  for i  = 1:n/PERBOX
    if vl1>4
      plot(X((i-1)*PERBOX+2:i*PERBOX,1),X((i-1)*PERBOX+2:i*PERBOX,2),'k-')
    else
      plot(X((i-1)*PERBOX+2:i*PERBOX,1),X((i-1)*PERBOX+2:i*PERBOX,2),'w-')
    end

    x1 = X((i-1)*PERBOX+1,1);
    x2 = X((i-1)*PERBOX+1,2);
    plot(x1,x2,'b.','MarkerSize',CENTRE_FSIZE*3.5);
    text(x1,x2,num2str(i),'FontSize',CENTRE_FSIZE,...
       'Color','w','HorizontalAlignment','Center');
  end
end
```

Computer-Based Decision Support System: Visual Mapping of Featured Database in Computer-Aided Diagnosis

Y. Wang[a], Z. Wang[a], L. Luo[a], S-H B. Lo[b], M. T. Freedman[b]

[a]Department of Electrical Engineering and Computer Science
The Catholic University of America, Washington, DC 20064, USA

[b]Department of Radiology and the Lombardi Cancer Center
Georgetown University Medical Center, Washington, DC 20007, USA

ABSTRACT

As a strategic move toward improving the utility of computer-aided diagnosis (CAD) in breast cancer detection, this work aims to develop a computer-based decision support system, through a visual mapping of featured database, to explain the entire decision making process jointly by the computer-encoded knowledge and the user-interaction. The main purpose of the work is twofold: enhance the clinical utility of CAD and provide a mechanism for optimal system design. We adopt a mathematical feature extraction procedure to construct the featured database from the suspicious mass sites localized by the enhanced segmentation. The optimal mapping of the data points is then obtained by learning a hierarchical normal mixtures and associated decision boundaries. A visual explanation of the decision making is further invented through a multivariate data mining and knowledge discovery scheme. In particular, using multiple finite normal mixture models and hierarchical visualization spaces, new strategy is that the top-level model and projection should explain the entire data set, best revealing the presence of clusters and relationships, while lower-level models and projections should display internal structure within individual clusters, such as the presence of subclusters, which might not be apparent in the higher-level models and projections. We demonstrate the principle of the approach on several multimodal numerical data sets, and we then apply the method to the visual explanation in CAD for breast cancer detection from digital mammograms.

1. INTRODUCTION

In order to improve mass detection and classification in clinical screening and/or diagnosis of breast cancers, many sophisticated computer-assisted diagnosis (CAD) systems have been recently developed. Although the clinical roles of the CAD systems may still be debatable, the fundamental role should be complementary to the radiologists' clinical duties or for automated high risk population screening. Literature survey has indicated that (1) most CAD systems are "black" boxes to the users and (2) no working link between "evaluation" and "improvement". This paper addresses the further development of CAD for mass detection based on (1) construction of featured knowledge database; (2) mapping of classified and unclassified data points; and (3) development of a visual exploration and explanation interface.

Although many previously proposed approaches have led to impressive results, several fundamental issues remain unresolved. For example, *Receiver Operating Characteristics* (ROC) analysis can provide an overall performance evaluation, it may not help the improvement of each of the multiple components in CAD system. Furthermore, since the machine observer and human observer may not detect the same set of masses, the "black box" nature of most CAD systems may prevent a natural on-line integration of human intelligence and further upgrade of a CAD system. Our effort is to: (1) provide a visual map of featured database before knowledge encoding component, so to evaluate and improve the pre-processing and signature extraction; (2) based on the map to design an optimal classifier best fitted to this particular database structure for knowledge encoding; and (3) combine the map, the classifier output, raw image, and user interface to explore and explain the whole decision making process by both radiologist and CAD systems.

Further author information: Send correspondence to Y. Wang (E-mail wang@pluto.ee.cua.edu).

2. BACKGROUND

As the first step toward understanding multivariate data sets, cluster information reveals insight that may prove useful in knowledge discovery since the growing volume of complex data are often high dimensional, multimodal, and lacking in prior knowledge..[4–6,9] Several new visualization methods have been progressively developed to model and display the contents of the data sets.[4,6–9,11,14] However, although such algorithms can usefully characterize the content of simple data sets, little comprehensive study has been reported that proves adequate in the face of multimodal and high dimensional data sets.[4,9,14] For example, a single projection of the data onto a visualization space may not be able to capture all of the interesting aspects of the data set. This motivates the consideration of a hierarchical visualization paradigm involving hierarchical statistical models and visualization spaces.

Once we explore the possibility of using many complementary visualization subspaces, cluster decomposition and dimensionality reduction are the two major steps. Cluster decomposition permits the use of relatively simple models for each of the local structures, offering greater ease of interpretation as well as the benefits of analytical and computational simplification. On the other hand, dimensionality reduction allows better visual interpretation and less computational demand. Many researchers have recently proposed various methods to improve data visualization.[6,9] The work most closely related to our methodology was reported by Bishop and Tipping in.[4,12] They introduce a hierarchical modeling and visualization algorithm based on a two-dimensional hierarchical mixture of latent variable models, whose parameters are estimated using the expectation-maximization (EM) algorithm.[4,19] The construction of the hierarchical tree proceeds top down in which the cluster decomposition is driven interactively by the user, and optimal projection is determined by maximum likelihood principle.

In this paper, we propose using standard finite normal mixtures (SFNM) and hierarchical visualization spaces for an effective data modeling and visualization. The strategy is that the top-level model and projection should explain the entire data set, best revealing the presence of clusters and relationships, while lower-level models and projections should display internal structure within individual clusters, such as the presence of subclusters, which might not be apparent in the higher-level models and projections. With many complementary mixture models and visualization projections, each level will be relatively simple while the complete hierarchy maintains overall flexibility yet still conveys considerable cluster information. Based on the concept of combining finite mixture modeling[19] and principal component projection[4,14] to guide cluster decomposition and dimensionality reduction, the particular advantages of our algorithm are:

1. At each level, a probabilistic principle component extraction is performed to project the softly partitioned data set down to a two-dimensional visualization space, leading to an effective dimensionality reduction, allowing effective separation and visualization of local clusters[4,8,15];
2. Learning from the data directly, information theoretic criteria are used to select model structures and estimate its parameter values, where the soft partitioning of the data set results in a standard finite normal mixture distribution best fitted to the data[7,21–25];
3. By alternatively performing principal component projection and finite mixture modeling, a complete hierarchy of complementary projections and refined models can be generated automatically, allowing a new paradigm of knowledge discovery.[4–6,9]

3. THEORY AND METHOD

One of the difficulties inherent in data visualization is the problem of visualizing multi-dimensionality.[4,6,9] When there are more than three variables, it stretches the imagination to visualize their relationships. Fortunately in data set with many variables, groups of variables often form clusters.[13,15,16] Thus, our approach includes two major complementary components: (1) dimensionality reduction by probabilistic principal component projection and (2) cluster decomposition by adaptive soft data clustering.

Assume the data points $\{\mathbf{t}_i\}$ in the data space come from K_0 clusters $\{\boldsymbol{\theta}_{\mathbf{t}1}, ..., \boldsymbol{\theta}_{\mathbf{t}k}, ..., \boldsymbol{\theta}_{\mathbf{t}K_0}\}$, where $\boldsymbol{\theta}_{\mathbf{t}k}$ is the Gaussian kernel parameter vector of cluster k in the model. Recently there has been considerable success in using the SFNM to model the distribution of a multimodal data set,[4,7,10,19,26] such that the data distribution takes a sum of the following general form:

$$p(\mathbf{t}) = \sum_{k=1}^{K_0} \pi_k g(\mathbf{t}|\boldsymbol{\theta}_{\mathbf{t}k}) \tag{1}$$

where π_k is the corresponding mixing proportion, with $0 \leq \pi_k \leq 1$ and $\sum \pi_k = 1$, and g is the Gaussian kernel. The problem of SFNM modeling addresses the combined estimation of regional parameters $(\pi_k, \boldsymbol{\theta}_{\mathbf{t}k})$ and detection of structural parameter K_0 in Eq. (1) based on the observations $\mathbf{t}$. One natural criterion used for estimating the parameter values is to minimize the distance between the SFNM distribution $f(\mathbf{t})$ and the data histogram $f_{\mathbf{t}}$. Suggested by information theory,[19,20] relative entropy (Kullback-Leibler distance) is a suitable measure, given by

$$D(f_{\mathbf{t}}\|f) = \sum_{\mathcal{T}} f_{\mathbf{t}}(\mathbf{t}) \log \frac{f_{\mathbf{t}}(\mathbf{t})}{f(\mathbf{t})}. \tag{2}$$

We have previously shown that distance minimization based on (2) is equivalent to the maximum likelihood (ML) estimation under a data independency approximation,[7] and when K_0 is given, the ML estimate of the regional parameters can be obtained using the EM algorithm.[15,19,26]

There are three major problems associated with the current approach. First, when the dimension of the data space is high, the computational complexity of implementing the EM algorithm in $\mathbf{t}$-space is very high. Second, the initialization of the EM algorithm is often heuristically chosen, which may lead to both local optima and computational complexity. Finally, since the number of the local clusters in a particular data set is generally unknown, model selection is a prerequisite. A natural way, with greater practical applicability, to tackle these problems is to introduce user interaction with the system.[4,9] Data mining and knowledge discovery are not processes that can be orchestrated a priori. Training algorithms and expected behavior can be specified, but the actual learning must follow for insight and spontaneous inspiration.[9] For example, by examining plots of principal component space, researchers often develop a deeper understanding of the driving forces that generated the original data, and effortlessly grasp the general characteristics of the data and propose an initial solution.[4,6,9]

Principal component analysis (PCA) is an effective method for achieving dimensionality reduction.[11,12] For a set of observed d-dimensional data vectors $\{\mathbf{t}_i\}$, $i \in \{1, ..., N\}$, the q principal axes $\mathbf{w}_m$, $m \in \{1, ..., q\}$, are those orthogonal axes onto which the retained variance under projection is maximal. It can be shown that the principal axes $\mathbf{w}_m$ are given by the q dominant eigenvectors (i.e., maximal eigenvalues) of the sample covariance matrix $\mathbf{C}_{\mathbf{t}} = \sum_i (\mathbf{t}_i - \boldsymbol{\mu}_{\mathbf{t}})(\mathbf{t}_i - \boldsymbol{\mu}_{\mathbf{t}})^{\mathrm{T}}/N$ such that $\mathbf{C}_{\mathbf{t}}\mathbf{w}_m = \lambda_m \mathbf{w}_m$ and where $\boldsymbol{\mu}_{\mathbf{t}}$ is the sample mean. The vector $\mathbf{x}_i = \mathbf{W}^{\mathrm{T}}(\mathbf{t}_i - \boldsymbol{\mu}_{\mathbf{t}})$, where $\mathbf{W} = (\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_q)$, is thus a q dimensional reduced representation of the observed vector $\mathbf{t}_i$. The advantage of PCA is twofold: the projection onto the principal subspace (1) minimizes the squared reconstruction error[12,15] and (2) maximizes the separation of data clusters.[16] Although the effectiveness of applying PCA in an unsupervised manner is highly data-dependent, our approach has a simple optimal appeal in that if the local clusters are linearly separable in a two- or three-dimensional space, the principal component projections allow best separation of the clusters.[16]

Suppose the data space is d-dimensional. Now consider a two-dimensional projection space $\mathbf{x} = (x_1, x_2)^{\mathrm{T}}$ together with a linear transformation, that maps the data space to the projection space by $\mathbf{x} = \mathbf{W}^{\mathrm{T}}(\mathbf{t} - \boldsymbol{\mu}_{\mathbf{t}})$ where $\mathbf{W}$ is a $d \times 2$ matrix. For a normal distribution $p(\mathbf{t})$ over the data space, using the rules of probability, a similar reduced dimension probability distribution of the new variables $\{\mathbf{x}_i\}$ in the projection space is obtained from the convolution of the projection model with the true distribution over data space in the form of $f(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{t})p(\mathbf{t})dt$.[4,12,17] Since the conditional distribution $p(\mathbf{x}|\mathbf{t}) = \delta(\mathbf{x} - \mathbf{W}^{\mathrm{T}}\mathbf{t} + \mathbf{W}^{\mathrm{T}}\boldsymbol{\mu}_{\mathbf{t}})$, where $\delta(.)$ is the delta function that $\delta(0) = 1$ and $\delta(\neq 0) = 0$, it can be shown that $f(\mathbf{x})$ is simply defined by the Radon transform of $p(\mathbf{t})$, i.e., $f(\mathbf{x}) = \int p(\mathbf{t})\delta(\mathbf{x} - \mathbf{W}^{\mathrm{T}}\mathbf{t} + \mathbf{W}^{\mathrm{T}}\boldsymbol{\mu}_{\mathbf{t}})dt$.[18] According to the linear superposition property of Radon transform and the projection invariant property of normal distribution, if $p(\mathbf{t})$ is a SFNM distribution, the data distribution in the projection space has a similar reduced dimension form as Eq. (1)

$$f(\mathbf{x}) = \sum_{k=1}^{K_0} \pi_k \int g(\mathbf{t}|\boldsymbol{\theta}_{\mathbf{t}k})\delta(\mathbf{x} - \mathbf{W}^{\mathrm{T}}\mathbf{t} + \mathbf{W}^{\mathrm{T}}\boldsymbol{\mu}_{\mathbf{t}})dt = \sum_{k=1}^{K_0} \pi_k g(\mathbf{x}|\boldsymbol{\theta}_{\mathbf{x}k}). \tag{3}$$

However, because of its global linearity, the application of PCA is necessarily somewhat limited.[12,13] For example, the inherent multimodal nature of the data set may be completely obscured when it is projected onto the lower dimensional principal subspace. Thus, it is important to note that although the cluster structure of the data set may be evident from the higher dimensional plot of the raw data, it is quite conceivable to have the intrinsic cluster structure of the data concealed after a projection in the more general case of high-dimensional data sets.[15] An alternative paradigm is to model multimodal data set with a collection of local linear subspaces through probabilistic principal component analysis as shown in Fig. 1.[12–14] The method is a two-stage procedure: a soft partitioning of the data space followed by estimation of the principal subspace within each partition. For the sake of computational simplicity, it is reasonable to consider the model parameter values being estimated firstly in the projection space and then further fine tuned in the data space.[14]

The association of a SFNM distribution with PCA offers the possibility of being able to visualize complex data structures through a mixture of probabilistic principal component subspaces. By a simple extension of the maximum a posterior for data classification in the standard K_0-ary Bayes hypothesis testing,[15,20] we can obtain a principal component projection along the desired axes onto which a particular portion of the data set is highlighted, by weighting all of the data points in the whole data set with their posterior probabilities belonging to that portion. This involves a soft clustering of the data points in which instead of any given data point being assigned exclusively to one principal component subspace, the responsibility for its generation is shared among all of the subspaces.

Under the SFNM model defined by Eq. (1), the posterior Bayesian probability z_{ik} of a given data point $\mathbf{t}_i$ belonging to cluster k is

$$z_{ik} = \frac{\pi_k g(\mathbf{t}_i|\boldsymbol{\theta}_{\mathbf{t}k})}{p(\mathbf{t}_i)}. \tag{4}$$

where $k = 1, 2, ..., K_0$ and $\sum_k z_{ik} = 1$. These posterior probabilities, together with the computational simplicity of performing PCA (involving no more than finding the top q eigenvectors of the covariance matrix of the data points) make it a good candidate for the linear subspace in the mixture. The q principal components define the local subspace assumed for the multimodal. The contributions of the input to the k subspace are the activities of the weighted data points $\{\mathbf{t}_{ik}\}$ for input cluster k. This can be obtained by $\mathbf{t}_{ik} = z_{ik}(\mathbf{t}_i - \boldsymbol{\mu}_{\mathbf{t}k})$, where $\boldsymbol{\mu}_{\mathbf{t}k}$ is the weighted sample mean of cluster k:

$$\boldsymbol{\mu}_{\mathbf{t}k} = \frac{\sum_i z_{ik}\mathbf{t}_i}{\sum_i z_{ik}}, \qquad \mathbf{C}_{\mathbf{t}k}=\frac{\sum_i z_{ik}(\mathbf{t}_i - \boldsymbol{\mu}_{\mathbf{t}k})(\mathbf{t}_i - \boldsymbol{\mu}_{\mathbf{t}k})^{\mathrm{T}}}{\sum_i z_{ik}} \tag{5}$$

The subspaces for the focused clusters are generated by a localized linear PCA such that $\mathbf{C}_{\mathbf{t}k}\mathbf{w}_{mk} = \lambda_{mk}\mathbf{w}_{mk}$. It is important to understand that each component in Eq. (1) now corresponds to an independent subspace model with parameters $\boldsymbol{\theta}_{\mathbf{x}k}$ and $\mathbf{W}_k$, where $\mathbf{W}_k= (\mathbf{w}_{1k}, \mathbf{w}_{2k}, ..., \mathbf{w}_{qk})$. More precisely, consider the vector $\mathbf{x}_{ik} = z_{ik}\mathbf{W}_k^{\mathrm{T}}(\mathbf{t}_i - \boldsymbol{\mu}_{\mathbf{t}k})$ to be a q dimensional reduced representation of k-cluster focused vector $\mathbf{t}_{ik}$, the corresponding probability distribution is defined by

$$g(\mathbf{x}|\mathbf{W}_k, \boldsymbol{\theta}_{\mathbf{x}k}) = \int g(\mathbf{t}|\boldsymbol{\theta}_{\mathbf{t}k})\delta(\mathbf{x} - \mathbf{W}_k^{\mathrm{T}}\mathbf{t} + \mathbf{W}_k^{\mathrm{T}}\boldsymbol{\mu}_{\mathbf{t}k})d\mathbf{t} \tag{6}$$

where the data mapping by $\mathbf{W}_k$ leads to an independent Radon transform. To interpret the corresponding set of visualization subspaces, it may be useful to plot all of the data points on every plot. For this, we may create a k-cluster focused projection in k-subspace by plotting the vector $\mathbf{x}_{ik}$, or display the density of "gray-level" in proportion to the contribution which each point has for k-subspace with $h[\mathbf{W}_k^{\mathrm{T}}(\mathbf{t}_i - \boldsymbol{\mu}_{\mathbf{t}k})] = z_{ik}$.

An important issue concerning unsupervised cluster decomposition is the detection of the structural parameter K_0, called model selection.[7,14,15,19,25] This is indeed particularly critical in real-world applications where the structure of the data patterns may be arbitrarily complex.[5] We propose to use two information theoretic criteria, i.e., the Akaike information criterion (AIC)[21] and minimum description length (MDL),[22] to guide model selection. The major thrust of this approach has been the formulation of a model fitting procedure in which an optimal model is selected from the several competing candidates such that the selected model best fits the observed data, under Jaynes' minimax entropy principle stated as "*the parameters in a model which determine the value of the maximum entropy should be assigned values which minimize the maximum entropy*".[23,24] For example, AIC tries to reformulate the problem explicitly as an *approximation* of the true structure by the model, implying that AIC will select the model that gives the minimum value defined by

$$\mathrm{AIC}(K_a) = -2\log(\mathcal{L}_{ML}) + 2K_a \tag{7}$$

where $\mathcal{L}_{ML}$ is the maximum likelihood of the model and K_a is the number of free adjustable parameters in the model. From a quite different point of view, MDL reformulates the problem explicitly as an information coding problem in which the best model fit is measured such that it assigns high probabilities to the observed data while at the same time the model itself is not too complex to describe.[22] A model is selected by minimizing the total description length defined by

$$\mathrm{MDL}(K_a) = -\log(\mathcal{L}_{ML}) + 0.5K_a \log N. \tag{8}$$

where the penalty term in MDL takes into account the number of observations. It should be pointed out that when the cluster separability is poor, the performance of these two information theoretic criteria may not be reliable.[21,25]

As discussed above, the SFNM model identification is first performed over **x**-space. However, a mapping from **t**-space to **x**-space may have the intrinsic cluster structure concealed, leading to an incorrect correspondence between Eq. (1) and Eq. (3). We now extend the mixture representation of Eq. (1) to form a hierarchical mixture model generally enough to be applicable to mixtures of any parametric density model. Based on the discussion of a two-level system consisting of a single Radon transform at the top level and a mixture of K_0 normal distributions at the second level, we can reformulate the hierarchy to a third level by associating a group $\mathcal{G}_k$ of SFNM models with each model k in the second level, given by

$$p(\mathbf{t}) = \sum_{k=1}^{K_0} \pi_k \sum_{j=1}^{L_{k,0}} \pi_{j|k} g(\mathbf{t}|\boldsymbol{\theta}_{\mathbf{t}(k,j)}) \tag{9}$$

where $\pi_{j|k}$ again correspond to a set of mixing proportions, one for each k, with $\sum_j \pi_{j|k} = 1$. The formation of the hierarchy is guided by the model selection over **x**-subspaces, where each level of the hierarchy corresponds to a generic model, with lower levels giving more focused and interpretable representations. Once again each component in Eq. (9) now corresponds to an independent subspace model with Radon transform $g(\mathbf{x}|\boldsymbol{\theta}_{\mathbf{x}(k,j)}) = \int g(\mathbf{t}|\boldsymbol{\theta}_{\mathbf{t}(k,j)})\delta(\mathbf{x} - \mathbf{W}^{\mathrm{T}}_{(k,j)}\mathbf{t} + \mathbf{W}^{\mathrm{T}}_{(k,j)}\boldsymbol{\mu}_{\mathbf{t}(k,j)})dt$.

4. ALGORITHMS

Based on the theory behind hierarchical mixtures of probabilistic principal component subspaces we have discussed above, we now present the description of our algorithm involving major steps of the visual hierarchy construction. Although the tree structure of the hierarchy may be empirically defined,[4,12] a more interesting effort, is to build the tree *automatically and interactively*. Guided by the two information theoretic criteria, our algorithm progressively proceeds by fitting a series of submodels to the clusters of the data set, in which model order is selected automatically and algorithm initialization is driven interactively. A schematic summary of the algorithm is as follows:

1. Project the data set onto a single **x**-space, in which **W** is determined from the sample covariance matrix $\mathbf{C_t}$ by fitting a single Gaussian model to the data set over **t**-space.
2. Learn $f(\mathbf{x})$ for $K = K_{MIN}, ..., K_{MAX}$, in which the values of π_k and $\boldsymbol{\theta}_{\mathbf{x}k}$ are initialized by the user and estimated by the EM algorithm over **x**-space.
3. Calculate the values of AIC and MDL for $K = K_{MIN}, ..., K_{MAX}$, and select a model with K_0 which corresponds to the minimum of AIC and MDL. The model parameters obtained in **x**-space will be used to initialize the model parameters in **t**-space for the learning in step 4.
4. Learn $f(\mathbf{t})$ with K_0, in which the values of π_k, z_{ik}, $\boldsymbol{\mu}_{\mathbf{t}k}$, and $\mathbf{C}_{\mathbf{t}k}$, are fine tuned by the EM algorithm over **t**-space..
5. Determine $\mathbf{W}_k$ from $\mathbf{t}_{ik}$ or $\mathbf{C}_{\mathbf{t}k}$, and plot $\mathbf{x}_{ik}$ or $h[\mathbf{W}_k^{\mathrm{T}}(\mathbf{t}_i - \boldsymbol{\mu}_{\mathbf{t}k})]$ onto **x**-subspaces at the second level for visual evaluation, for $k = 1, 2, ..., K_0$.
6. Learn $\mathcal{G}_k(\mathbf{t})$ by repeating steps $2-4$ and construct **x**-subspaces at the third level by repeating step 5, for $k = 1, 2, ..., K_0$.
7. Complete the whole hierarchy under the information theoretic criteria, and plot all **x**-subspaces for visual exploration and explanation.

Our algorithm begins by determining $\mathbf{W}$ for the top level projection. For low dimensional data sets, we directly evaluate the covariance matrix $\mathbf{C_t}$ to find $\mathbf{W}$.[13,15] For high dimensional cases, since only the top two eigenvectors of the covariance matrix of the data points are of the interest, it may be computationally more efficient to apply our previously developed APEX neural networks[8] to find $\mathbf{W}$ directly from the data points $\mathbf{t}_i$ (Step 1). On the basis of this single $\mathbf{x}$-space, given a fixed K, the user then selects (K_{MIN}, K_{MAX}) and points $\boldsymbol{\mu}_{\mathbf{x}k}$ on the plot corresponding to the centers of apparent clusters. The EM algorithm can be applied to allow a SFNM (Eq. (3)) to be fitted to the projected data through the following two-stage[19,26] form:

E-Step

$$z_{ik}^{(n)} = \frac{\pi_k^{(n)} g(\mathbf{x}_i|\theta_{\mathbf{x}k}^{(n)})}{f(\mathbf{x}_i|\pi_k^{(n)}, \theta_{\mathbf{x}k}^{(n)})} \tag{10}$$

M-Step

$$\pi_k^{(n+1)} = \frac{1}{N}\sum_{i=1}^{N} z_{ik}^{(n)}, \quad \boldsymbol{\mu}_{\mathbf{x}k}^{(n+1)} = \frac{\sum_{i=1}^N z_{ik}^{(n)}\mathbf{x}_i}{\sum_{i=1}^N z_{ik}^{(n)}}, \quad \mathbf{C}_{\mathbf{x}k}^{(n+1)} = \frac{\sum_{i=1}^N z_{ik}^{(n)}(\mathbf{x}_i - \boldsymbol{\mu}_{\mathbf{x}k}^{(n)})(\mathbf{x}_i - \boldsymbol{\mu}_{\mathbf{x}k}^{(n)})^{\mathrm{T}}}{\sum_{i=1}^N z_{ik}^{(n)}} \tag{11}$$

where at each complete cycle of the algorithm, we first use "old" set of parameter values to determine the posterior probabilities $z_{ik}^{(n)}$ using Eq. (10). These posterior probabilities are then used to obtain "new" values $\pi_k^{(n+1)}$, $\boldsymbol{\mu}_{\mathbf{x}k}^{(n+1)}$, and $\mathbf{C}_{\mathbf{x}k}^{(n+1)}$ using Eqs.(11). The algorithm cycles back and forth until the value of relative entropy (Eq. (2)) reaches its minimum (Step-2). It can be shown that, at each stage of the EM algorithm, the relative entropy decreases unless it is already at a local minimum.[19] The model selection procedure will then determine the optimal number K_0 of models to fit at the next level down using the two information theoretic criteria, where $K_a = 6K_0 - 1$ including $2K_0$ means, $2K_0$ variances, K_0 correlation coefficients, and $K_0 - 1$ mixing factors (Step 3). The resulting points $\boldsymbol{\mu}_{\mathbf{t}k}^{(0)}$ in data space, obtained by $\boldsymbol{\mu}_{\mathbf{t}k}^{(0)} = \mathbf{W}\boldsymbol{\mu}_{\mathbf{x}k}^{(\infty)} + \boldsymbol{\mu}_{\mathbf{t}}$, are then used as the initial means of the respective submodels. Since the mixing proportions π_k are projection-invariant, we simply assign a 2×2 unit matrix to the remaining parameters of the covariance matrix $\mathbf{C}_{\mathbf{t}k}$. Once again the EM algorithm can be applied to allow a SFNM (Eq. (1)) with K_0 submodels to be fitted to the data over $\mathbf{t}$-space. In order to obviate the need to store all the incoming observations, and change the parameters immediately after each data point, it may be computationally more efficient to apply our previously developed probabilistic self-organizing map (PSOM), an incremental EM algorithm,[7] to estimate $p(\mathbf{t})$.

With a soft partitioning of the data set using the PSOM, data points will now effectively belong to more than one cluster at any given level. Thus, the effective input values are $\mathbf{t}_{ik} = z_{ik}(\mathbf{t}_i - \boldsymbol{\mu}_{\mathbf{t}k})$ for an independent visualization subspace k in the hierarchy. We then extend our APEX algorithm to a probabilistic version, i.e., PAPEX,[8,27] to determine $\mathbf{W}_k$, summarized as follows (Step 4).

1. Initialize the feedforward weight vector $\mathbf{w}_{mk}$ for $m = 1, 2$, and the feedback weight vector a_k to small random values at time $i = 1$. Assign a small positive value to the learning rate parameter η.
2. Set $m = 1$, and for $i = 1, 2, ...$, compute
$$y_{1k}(i) = \mathbf{w}_{1k}^{\mathrm{T}}(i) z_{ik}(\mathbf{t}_i - \boldsymbol{\mu}_{\mathbf{t}k}), \quad \mathbf{w}_{1k}(i+1) = \mathbf{w}_{1k}(i) + \eta[y_{1k}(i) z_{ik}(\mathbf{t}_i - \boldsymbol{\mu}_{\mathbf{t}k}) - y_{1k}^2(i)\mathbf{w}_{1k}(i)] \tag{12}$$
For large i we have $\mathbf{w}_{1k}(i) \longrightarrow \mathbf{w}_{1k}$, where $\mathbf{w}_{1k}$ is the eigenvector associated with the largest eigenvalue of the covariance matrix $\mathbf{C}_k$.
3. Set $m = 2$, and for $i = 1, 2, ...$, compute
$$y_{2k}(i) = \mathbf{w}_{2k}^{\mathrm{T}}(i) z_{ik}(\mathbf{t}_i - \boldsymbol{\mu}_{\mathbf{t}k}) + a_k(i) y_{1k}(i), \quad \mathbf{w}_{2k}(i+1) = \mathbf{w}_{2k}(i) + \eta[y_{2k}(i) z_{ik}(\mathbf{t}_i - \boldsymbol{\mu}_{\mathbf{t}k}) - y_{2k}^2(i)\mathbf{w}_{2k}(i)] \tag{13}$$
$$a_k(i+1) = a_k(i) - \eta[y_{2k}(i) y_{1k}(i) + y_{2k}^2(i) a_k(i)] \tag{14}$$
For large i we have $\mathbf{w}_{2k}(i) \longrightarrow \mathbf{w}_{2k}$, where $\mathbf{w}_{2k}$ is the eigenvector associated with the second largest eigenvalue of the covariance matrix $\mathbf{C}_k$.

Having determined principal axes $\mathbf{W}_k$ of the mixture model at the second level, we will construct the visualization subspaces by plotting each data point $\mathbf{t}_i$ at the corresponding $\mathbf{x}_{ik}$. Thus if one particular point takes most of the contribution for a particular component, then that point will effectively be visible only on the corresponding subspace (Step 5).

Determination of the parameters of the models at the third level can again be viewed as a two-step estimation problem, in which further split of the models at the second level is determined within each of the subspaces over $\mathbf{x}$-space, and then the parameters of the selected models are fine tuned over $\mathbf{t}$-space. Similarly, the resulting model estimated over $\mathbf{x}$-space are then used to initialize the means of the respective submodels over $\mathbf{t}$-space. The corresponding $\mathcal{G}_k(\mathbf{t})$ can again be estimated using the EM or PSOM algorithm[7,19,26] to allow a SFNM distribution with $L_{k,0}$ submodels to be fitted to the data. In the E-step, the posterior probability that data point $\mathbf{t}_i$ belongs to submodel j is given by

$$z_{i(k,j)} = z_{ik} z_{ij|k} = z_{ik} \frac{\pi_{j|k} g(\mathbf{t}_i|\boldsymbol{\theta}_{\mathbf{t}j|k})}{\mathcal{G}(\mathbf{t}_i|\boldsymbol{\theta}_{\mathbf{t}k})}. \tag{15}$$

where z_{ik} are constants estimated from the second level of the hierarchy. The corresponding M-step includes

$$\pi_{j|k} = \frac{\sum_{i=1}^{N} z_{i(k,j)}}{\sum_{i=1}^{N} z_{ik}}, \quad \boldsymbol{\mu}_{\mathbf{t}(k,j)} = \frac{\sum_{i=1}^{N} z_{i(k,j)} \mathbf{t}_i}{\sum_{i=1}^{N} z_{i(k,j)}}, \quad \mathbf{C}_{\mathbf{t}(k,j)} = \frac{\sum_{i=1}^{N} z_{i(k,j)} (\mathbf{t}_i - \boldsymbol{\mu}_{\mathbf{t}(k,j)})(\mathbf{t}_i - \boldsymbol{\mu}_{\mathbf{t}(k,j)})^{\mathrm{T}}}{\sum_{i=1}^{N} z_{i(k,j)}}. \tag{16}$$

With the resulting $z_{i(k,j)}$ in $\mathbf{t}$-space, we can apply the PAPEX algorithm to estimate $\mathbf{W}_{(k,j)}$, in which the effective input values are expressed by $\mathbf{t}_{i(k,j)} = z_{i(k,j)}(\mathbf{t}_i - \boldsymbol{\mu}_{\mathbf{t}(k,j)})$. The next level visualization subspace is generated by plotting each data point $\mathbf{t}_i$ at the corresponding $\mathbf{x}_{i(k,j)} = z_{i(k,j)} \mathbf{W}_{(k,j)}^{\mathrm{T}} (\mathbf{t}_i - \boldsymbol{\mu}_{\mathbf{t}(k,j)})$ in (k,j)-subspace (Step 6).

The construction of the entire tree structure hierarchy is automatically completed when no further data split is recommended by the information theoretic criteria in all of the parent subspaces (Step 7).

5. ILLUSTRATION AND APPLICATION

We first illustrate the application of our algorithm to a simple synthetic data set. Fig. 1 (a) shows a data set consisting of 450 data points generated from a mixture of three Gaussians in three-dimensional space. Each Gaussian is relatively flat (has small variance) in one dimension. Two of these pancake-like clusters are closely spaced, while the third is well separated from the first two. The dimensionality of this data set has been chosen to illustrate the basic principle of the approach. The global view of the raw data over $\mathbf{t}$-space clearly suggests the presence of three distinct clusters within the data.

To explore the data characteristics, we first perform a single global PCA to project each data point onto a single $\mathbf{x}$-space (top level), shown in Fig. 1 (b). Both the user inspection and the two information theoretic criteria have clearly suggested the presence of two distinct clusters within the projected data set. Based on a soft clustering of the data points, we then apply PAPEX to both clusters and generate the two corresponding independent cluster-focused subspaces (second level), as shown in Fig. 1 (c). Not to our surprise, the two information theoretic criteria have suggested a further split of cluster 2 but not of cluster 1. Once again by performing three independent PAPEX, the final cluster decomposition through the cluster-focused subspaces (third level) is completed shown in Fig. 1 (d).

With this three-level hierarchical data exploration, the capable nature of the approach is evident as the interim two subspaces (second level) only attempt to highlight the data points which have already been modeled by their immediate ancestor (top level). Indeed, the model fitting procedure has successfully discovered all three data clusters. The original data clusters have been individually colored, and it can be seen that the red, yellow, and blue data points have been well separated and highlighted in the third level subspaces.

As an example of a more complex problem, we consider a data set arising from a mixture of three closely spaced Gaussians consisting of 300 data points, shown in Fig. 2 (a). Once again the original data clusters have been individually colored. We first apply APEX to extract the global principal axis, indicated by the black line in Fig. 2 (a). The two information theoretic criteria have suggested the presence of three distinct clusters, where the user then selects three initial cluster centers and the EM/PSOM algorithm is applied to perform a soft clustering of the data points. This leads to a mixture of three independent probabilistic principal component subspaces whose principal axes are separately extracted, indicated by the yellow lines in Fig. 2 (a). The contributions of each data point to these subspaces, in terms of its "gray-level" $h[\mathbf{t}_i] = z_{ik}$, are displayed over $\mathbf{t}$-space in Fig. 2 (b).

Since the model selection and algorithm initialization are performed over x-space with user's interaction, it may be helpful to investigate the visual effectiveness of dimensionality reduction using the probabilistic principal component projections.[4,9] Based on the estimated $\mathbf{W}_k$, we have constructed each of the cluster-focused subspaces using both "data graphics" (e.g., in terms of $\mathbf{x}_{ik} = z_{ik}\mathbf{W}_k^{\mathrm{T}}(\mathbf{t}_i - \boldsymbol{\mu}_{tk})$) and "data image" (e.g., in terms of $h[\mathbf{W}_k^{\mathrm{T}}(\mathbf{t}_i - \boldsymbol{\mu}_{tk})] = z_{ik}$) techniques. As a more overlapped case, Fig. 2 (c-d) present the plots of "data graphics" and "data image" from the data set, where "data graphics" emphasizes the contribution of a particular data point to that particular subspace concerning its geometric distance to the center of the cluster, while "data image" emphasizes the effectiveness of a data point reflecting its global appearance. It can be seen that the plot of each cluster is clean and well-shaped.

In order to quantitatively evaluate the effectiveness of our approach with user interactions,[9] we apply our algorithm to a synthesized testing data set given in Fig. 3 (up-left). Using the APEX algorithm we accurately estimate the top global principal axis, indicated by the back line. By projecting the data points onto a two-dimensional x-space, all three data clusters are visible. This plot indicates that although the second advantage of PCA forementioned is highly data-dependent, when the data clusters are linearly separable in a projection space, the principal component projections allow effective separation of the clusters.[16] We then apply the two information theoretic criteria to examine this plots. In this case, we set $K_{MIN} = 1$ and $K_{MAX} = 5$. The minima of both AIC and MDL have clearly suggested a three-cluster data structure, as given by the curve in Fig. 3 (third block in the second row). Thus a two-level SFNM model may be sufficient. We then conduct two experiments to assess the performance of our algorithm. Since all the model parameters are known in this case, the true top principal axes of the data clusters have been individually calculated. First, we compare the estimated top principal axes of the data clusters using our algorithm with the corresponding true top principal axes. From the down-right block in Fig. 3, it can be seen that the two sets of the top principal axes are perfectly matched (blue lines). Second, we use the global relative entropy (GRE) between the data histogram and the estimated SFNM model to measure the goodness of model fitting. The numerical result through our experiments indicates a very good performance with a GRE value of 0.008 nats.

User interaction with the algorithm is an important issue. We have developed a user-friendly graphical interface to facilitate the data visualization purpose, as shown in Fig. 3. By allowing the user to select the initial centers of the data clusters demonstrated in Fig. 3, our experience has convincingly indicated a great reduction of both computational complexity and local optimum likelihood. For example, compared to the results of model selection reported by Akaike[21] and Wax,[25] the curves of the AIC and MDL generated by our algorithm are much more consistent and smooth, and user-initialized computation is five times (in average) faster than the random trials. It should be pointed out that although the final SFNM model can be estimated, the pathways of achieving cluster decomposition may be multiple. For example, in this case the user has the flexibility to select only two clusters in the second level and to further split the "right" cluster, thus to adopt a three-level hierarchy. We believe that this user-driven nature of the current algorithm is also highly appropriate for the visualization context.[4,14]

Since a more convincing example should involve more clusters with multiple levels, we have also applied our algorithm to the same data set used by Bishop&Tipping,[4] shown in Fig. 4 (a). This data set arises from a noninvasive monitoring system used to determine the quantity of oil in a multiphase pipeline containing a mixture of oil, water, and gas.[4] The experiment gives 12 diagnostic measurements in total. Our interim goal is to visualize the structure of the data in the original 12-dimensional space. A data set consisting of 1,000 points is obtained synthetically and the data is expected to have an intrinsic dimensionality of two corresponding to the two dominant components (e.g., oil and water). However, the presence of different flow configurations leads to numerous distinct clusters. We then apply our algorithm to perform a cluster discovery. Results from partially fitting the oil flow data using a three-level hierarchical model are given in Fig. 4. It should be pointed out that since the "right" answer to this real-world data set is not available, we are not able to validate this new result. However, we believe that this example has clearly been highly successful, note how the selected single cluster (number 2) in the top-level plot, is discovered to be two quite separated clusters at the second level.

As a final example, we consider the visual explanation in computer-aided diagnosis (CAD) for breast cancer detection. As a step toward improving the performance of CAD system, we have put considerable efforts to conduct various studies and develop reliable image enhancement and lesion segmentation techniques.[7] More precisely, we try to make both the hidden data patterns and the neural network "black box" to be as transparent as possible to the user (e.g., radiologists and patients) through interactive visual explanation. The clinical goal is to eliminate the false positive sites that correspond to normal dense tissues with mass-like appearances through featured discrimination. We adopt a mathematical feature extraction procedure to construct our database from all the suspicious mass sites localized by the enhanced segmentation.[7] The optimal mapping of the data points is then obtained by learning the generalized normal mixtures and decision boundaries, where a probabilistic modular neural network is developed to carry out both soft and hard clustering.[7] The joint histogram of the featured database extracted from true and false mass regions are investigated and the features that can better separate the true and false mass sites are selected. Our experience has suggested that three imagery features, i.e., site area, compactness, and difference entropy, were having good discrimination and reliability properties.

We then use our previously developed algorithm[7] to distinguish the true masses from false masses based on the features extracted from the suspected regions. 150 mammograms were selected from the mammogram database. Each mammogram contained at least one mass case of varying size and location. The areas of suspicious masses were identified following the proposed procedure with biopsy proven results. In a typical experiment, we have selected a three-dimensional feature space consisting of compactness I, compactness II, and difference entropy. It should be noticed that the feature vector can easily extend to higher dimensionality. A training feature vector set was constructed from 50 true mass ROIs and 50 false mass ROIs, where ROI stands for *region of the interest.* In addition to the decision boundaries recommended by the computer algorithms, a visual explanation interface has also been integrated with hierarchical projections. Fig. 5 (a) shows the database map selection with compactness definition I and difference entropy. Fig. 5 (b) shows the database map selection with compactness definition II and difference entropy. Our experience has suggested that the recognition rate with compactness I are more reliable than that with compactness II.

We have conducted a preliminary study to evaluate the performance of the algorithms in real case detection, in which 6 – 15 suspected masses per mammogram were detected and required further clinical decision making. We found that the proposed visual explanation approach, together with CAD system, can reduce the number of suspicious masses with a sensitivity of 84% at a specificity of 82% (1.6 false positive findings per mammogram) based on the database containing 46 mammograms (23 of them have biopsy proven masses). Fig. 6 shows a representative mass detection result on one mammogram with a stellate mass, indicated by the arrow in Fig. 6 (a). After appropriate feature extraction, ten sites with brightest intensity were selected, shown in Fig. 6 (b). The featured vectors of these candidates were submitted against the estimated "probability cloud" for visual explanation as a decision support, together with the opinion recommended by our CAD system. The final results indicated that the stellate mass lesion was correctly detected, confirmed by our experience radiologists, shown in Fig. 6 (c). It should be pointed out that in this real-world application, a higher recognition rate may be controlled by the domain experts in balancing the trade-off between the *false positive* and *false negative* rates.[7]

6. DISCUSSION

We have presented a novel approach to visual explanation for data mining and knowledge discovery, which is both statistically principled and visually effective. This method, as illustrated by the well-planned simulations and pilot applications in computer-aided diagnosis, can be very capable of revealing hidden structure within data. It is important to emphasize that in relation to previous work,[4,11–13] one interesting consideration with the present algorithm is that the models are determined by the information theoretic criteria, and this criterion can not only select the most appropriate model structure but also allow an user-driven portfolio as a double check. This approach promotes a self-consistent fitting of the whole tree, so that an automated procedure for generating the hierarchy becomes reality.[4] In addition, since we perform model selection and parameter initialization firstly over the projection space, the computational complexity is greatly reduced in compared to the maximum likelihood estimation in full dimension. Our case study of a seven dimensional data set has indicated at least a 50% reduction of the computational time. Other possible advantages include the determination of data projection by maximum the separation of clusters which in turn optimizes the other crucial operations such as model selection and parameter initialization,[16] and data rendering algorithms which permit user or hypothesis driven nature of the data projection.[14]

Acknowledgment

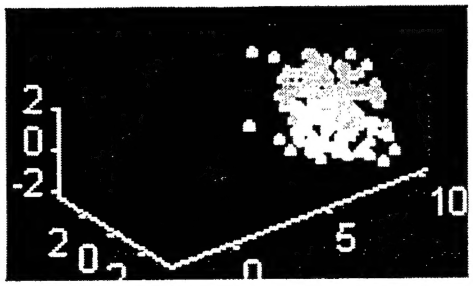
This work is supported by the Department of Defense under Grants DAMD17-98-1-8045 and DAMD17-98-1-8044.

REFERENCES

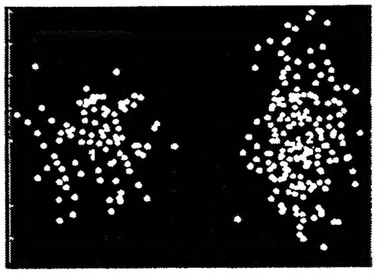
1. S. M. Lai, X. Li, and W. F. Bischof, "On Techniques for Detecting Circumscribed Masses in Mammograms," *IEEE Trans. on Med. Imaging,* Vol. 8, No. 4, pp. 377-386, 1989.

2. H. P. Chan, D. Wei, M. A. Helvie, B. Sahiner, D. D. Alder, M. M. Goodsitt, and N. Petrick, "Computer-Aided Classification of Mammographic Masses and Normal Tissue: Linear Discriminant Analysis in Texture Feature Space," *Phys. Med. Biol.*, Vol. 40, pp. 857-876, 1995.
3. R. M. Haralick, K. Shanmugam, and I. Dinstein, "Textural Features for Image Classification," *IEEE Trans. on Sys., Man, and Cyber.*, Vol. SMC-3, No. 6, pp. 610-621, Nov. 1973.
4. C. M. Bishop and M. E. Tipping, "A hierarchical latent variable model for data visualization," *IEEE Trans. Pattern Anal. Machine Intell.*, Vol. 20, No. 3, pp. 282-293, March 1998.
5. T. R. Golub, D. K. Slonim, *et al.*, "Molecular classification of cancer: class discovery and class prediction by gene expression monitoring," *Science*, Vol. 286, pp. 531-537, October 1999.
6. E. R. Tufte, *Visual Explanation: Images and Quantities, Evidence and Narrative*, Graphics Press, Cheshire 1996.
7. Y. Wang, S. H. Lin, H. Li, and S, Y. Kung, "Data mapping by probabilistic modular networks and information theoretic criteria," *IEEE Trans. Signal Processing*, Vol. 46, No.12, pp. 3378-3397, December 1998.
8. S. Y. Kung, *Principal Component Neural Networks*, New York: John Wiley, 1996.
9. G. M. Nielson, "Challenges in visualization research," *IEEE Trans. Visualization and Computer Graphics*, vol. 2, no. 2, pp. 97-99, June 1996.
10. M. I. Jordan and R. A. Jacobs, "Hierarchical mixture of experts and the EM algorithm," *Neural Computation*, Vol. 6, pp. 181-214, 1994.
11. N. Kambhatla and T. K. Leen, "Dimension reduction by local principal component analysis," *Neural Computation*, Vol. 9, No. 7, pp. 1493-1516, 1997.
12. M. E. Tipping and C. M. Bishop, "Mixtures of probabilistic principal component analyzers," *Neural Computation*, Vol. 11, pp. 443-482, 1999.
13. G. E. Hinton, P. Dayan, and M. Revow, "Modeling the manifolds of images of handwritten digits," *IEEE Trans. Neural Net.*, Vol. 8, No. 1, pp. 65-74, January 1997.
14. L. Luo, Y. Wang, and S. Y. Kung, "Hierarchy of probabilistic principal component subspaces for data mining," *Proc. IEEE Workshop on Neural Nets for Signal Processing*, Wisconsin, August 1999.
15. S. Haykin, *Neural Networks: A Comprehensive Foundation,* 2nd ed., Prentice Hall, Inc., Upper Saddle River, Ney Jersey, 1999.
16. K. Etemad and R. Chellappa, "Separability-based multiscale basis selection and feature extraction for signal and image classification," *IEEE Trans. Image Processing*, Vol. 7, No. 10, October 1998.
17. R. Gray and L. Davisson, *Random Processes–A Mathematical Approach for Engineers*, Englewood Cliffs, NJ: Prentice-Hall, Inc. 1986.
18. R. N. Bracewell, *Two-Dimensional Imaging,* Prentice-Hall, Inc., 1995.
19. D. M. Titterington, A. F. M. Smith, and U. E. Markov, *Statistical analysis of finite mixture distributions.* New York: John Wiley, 1985.
20. T. M. Cover and J. A. Thomas, *Elements of Information Theory*, New York: Wiley, 1991.
21. H. Akaike, "A new look at the statistical model identification," *IEEE Trans. Automatic Control*, Vol. 19, No. 6, pp. 716-723, 1974.
22. J. Rissanen, "Modeling by shortest data description," *Automat.*, Vol. 14, pp. 465-471, 1978.
23. E. T. Jaynes, "Information theory and statistical mechanics," *Phys. Rev.*, Vol. 106, No. 4, pp. 620-630/171-190, May 1957.
24. J. Rissanen, "Minimax entropy estimation of models for vector processes," *System Identification: Advances and Case Studies*, pp. 97-117, Academic Press, 1987.
25. M. Wax and T. Kailath, "Detection of Signals by Information Theoretic Criteria," *IEEE Trans. Acoust. Speech, Signal Processing,* Vol. 33, No. 2, April 1985.
26. L. I. Perlovsky and M. M. McManus, "Maximum likelihood neural networks for sensor fusion and adaptive classification," *Neural Networks*, Vol. 4, pp. 89-102, 1991.
27. S. Y. Kung, K. I. Diamantaras, and J. S. Taur, "Adaptive Principal Component Extraction (APEX) and Applications," *IEEE Trans. Signal Processing*, Vol. 42, No. 5, pp. 1202-1217, May 1994.

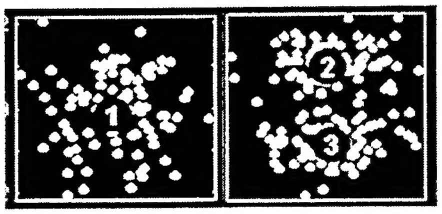
Figure 1



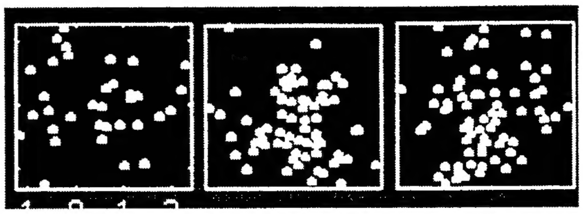
(a)



(b)

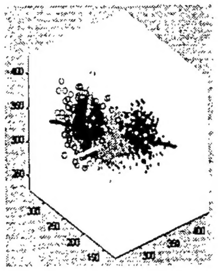


(c)

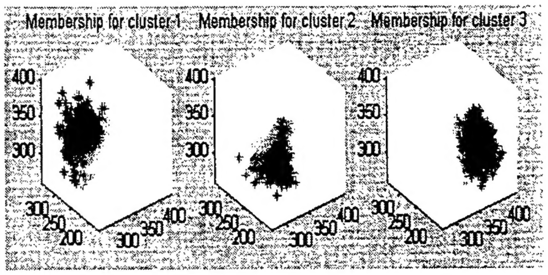


(d)

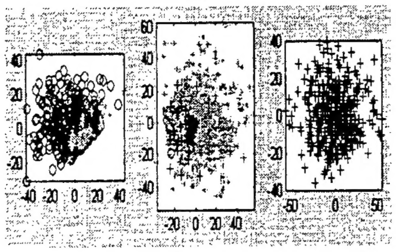
Figure 2



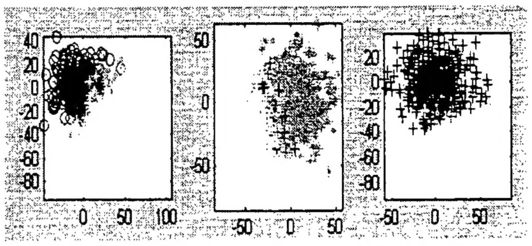
(a)



(b)

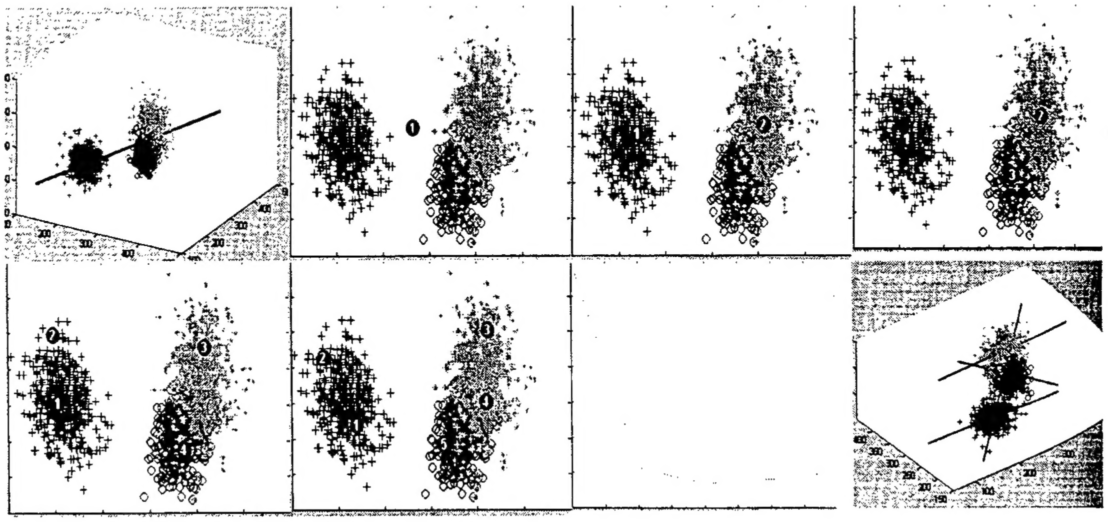


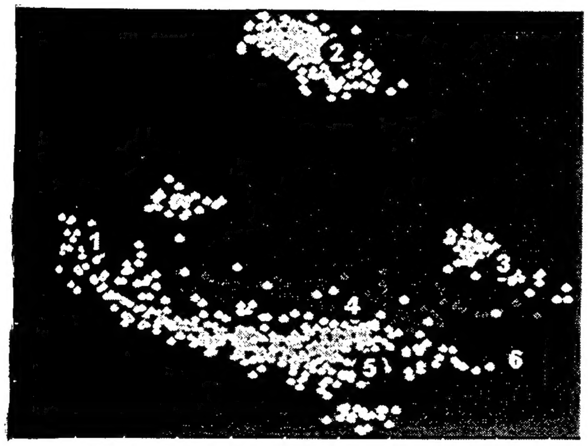
(c)



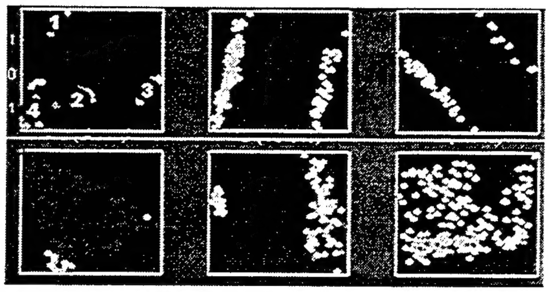
(d)

Figure 3.





(b)



(c)

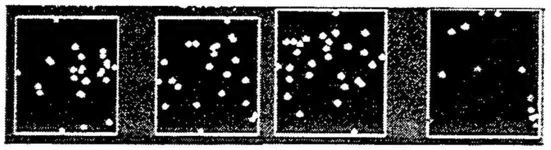
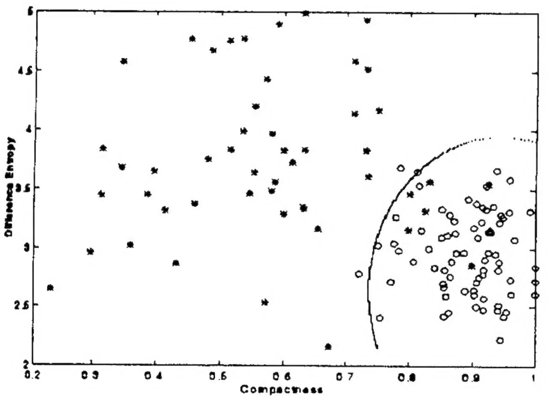


Figure 4. (a)



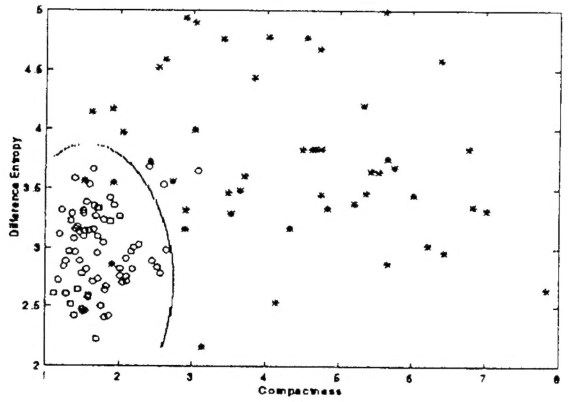
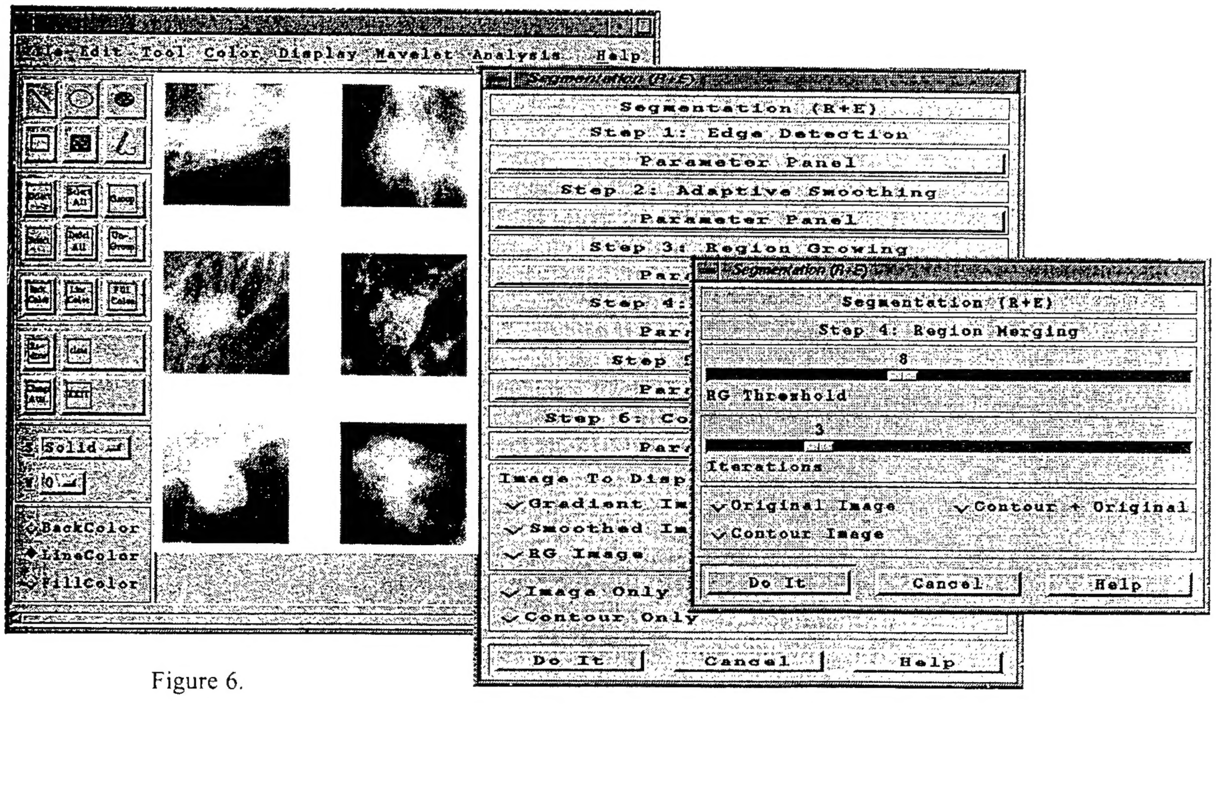


Figure 5. (a) (b)



Segmentation (R+E)
Step 1: Edge Detection
Parameter Panel
Step 2: Adaptive Smoothing
Parameter Panel
Step 3: Region Growing
Step 4: Region Merging
Step 6:
Image To Display
✓ Gradient Image
✓ Smoothed Image
✓ RG Image
✓ Image Only
✓ Contour Only

RG Threshold 8
Iterations 3
✓ Original Image
✓ Contour + Original
✓ Contour Image

Figure 6.